| | |
|---|---|
| **Titre:** Title: | Analyse mathématique, méthode de calcul de la gigue et applications aux réseaux Internet |
| **Auteur:** Author: | Hadhami Dbira |
| **Date:** | 2017 |
| **Type:** | Mémoire ou thèse / Dissertation or Thesis |
| **Référence:** Citation: | Dbira, H. (2017). Analyse mathématique, méthode de calcul de la gigue et applications aux réseaux Internet [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/2463/ |



| | |
|---|---|
| **URL de PolyPublie:** PolyPublie URL: | https://publications.polymtl.ca/2463/ |
| **Directeurs de recherche:** Advisors: | Brunilde Sanso, & André Girard |
| **Programme:** Program: | Génie électrique |

UNIVERSITÉ DE MONTRÉAL

# ANALYSE MATHÉMATIQUE, MÉTHODE DE CALCUL DE LA GIGUE ET APPLICATIONS AUX RÉSEAUX INTERNET

HADHAMI DBIRA
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(GÉNIE ÉLECTRIQUE)
JANVIER 2017

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

# ANALYSE MATHÉMATIQUE, MÉTHODE DE CALCUL DE LA GIGUE ET APPLICATIONS AUX RÉSEAUX INTERNET

présentée par : DBIRA Hadhami
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. FRIGON Jean-François, Ph. D., président
Mme SANSO Brunilde, Ph. D., membre et directrice de recherche
M. GIRARD André, Ph. D., membre et codirecteur de recherche
M. LE NY Jerôme, Ph. D., membre
Mme GRAVEY Annie, Ph. D., membre externe

# DÉDICACE

*À ma famille,*
*et à tous mes ami(e)s...*

# REMERCIEMENTS

Sans aucun doute, ce travail de recherche n'aurait pu pas être mené à bien sans le soutien substantiel, les commentaires constructifs et les précieux conseils de mes directeurs de recherche les professeurs Brunilde Sansò et André Girard. Je leur exprime ma profonde gratitude et mon total respect pour la confiance accordée, leur disponibilité et leurs soutiens moral et financier.

Je remercie également les professeurs Jean-François Frigon, Annie Gravey et Jerôme Le Ny d'avoir accepté de prendre part au jury de ma soutenance.

Que le professeur Antonio Capone de Polytechnique de Milan puisse trouver ici l'expression de ma gratitude pour son aimable collaboration dans la réalisation des différents tests expérimentaux.

Mes vifs remerciements s'adressent à Hakim Mellah, Hamza Dahmouni et Mohamed Ouzineb pour leurs conseils lucides et pour leur support.

Si j'ai pu aller au bout de mon doctorat, c'est aussi grâce à la bonne ambiance que j'ai trouvée au laboratoire. Raison pour laquelle, je tiens à remercier mes collègues au LORLAB et au GERAD désormais des amis proches : Adham, Arash, Carol, Carmelo, Federico, Filippo, Lê, Luca, Marnie, Mickel, Samuel, et Silvia. Un grand merci pour vos précieux conseils et votre soutien inconditionnel.

Mes remerciements vont aussi aux personnel du département génie électrique de Polytechnique Montréal et du GERAD, Marie Perreault, Carole Dufour, Francine Benoit, Karine Hébert, Marilyne Lavoie, Edoh Liagros Logo, Pierre Girard, Nathalie Lévesque, Ginette Desparois, Rachel Lortie et Jean Bouchard.

Ce remerciement est en outre adressé à Nader Meddeb pour avoir accepté de lire et de commenter le texte de la présente recherche

Finalement, la liste serait incomplète sans ma famille, qui m'a été d'un inestimable soutien durant cette aventure et mes chers amis devenus ma deuxième famille : Amal, Foued, Sabrine, Nesrine, Ghaya, Aymen et Hamza, pour leurs encouragements pendant les moments difficiles.

# RÉSUMÉ

Internet, ces dernières années, sert de support de communication à un grand nombre d'applications. L'évolution des réseaux à haut débit ont facilité le progrès des applications multimédia comme la voix sur IP, la vidéo streaming ou la vidéo interactive en temps réel... La variation de la disponibilité des ressources du réseau ne peut pas garantir une bonne qualité à tout moment pour ces services. C'est dans ce contexte que les travaux de ce projet de doctorat s'inscrivent et précisément dans le cadre de l'optimisation de la qualité de service (QoS). Les mécanismes de contrôle de QoS sont variés. On retrouve le contrôle de délai, assuré par la stratégie d'ordonnancement des paquets. Le contrôle de débit, quant à lui, fait en sorte que le débit de la source soit égal à la bande passante disponible dans le réseau. Excepté que les applications vidéo, surtout en temps réel, sont très sensibles à la variation du délai, appelée *la gigue*. En effet, la qualité perçue par les clients des vidéos en ligne dépend étroitement de la gigue. Une augmentation de la gigue engendre principalement des problèmes de démarrage retardé de la vidéo, des interruptions au cours de la vidéo et des distorsions de la résolution.

L'objectif de cette thèse est d'étudier le paramètre de la gigue, qui demeure peu étudiée dans la littérature sur les réseaux IP, ainsi que d'envisager l'impact de l'augmentation de ce paramètre sur la vidéo transmise sur IP, l'une des applications les plus populaires de nos jours. Toutefois, au-delà des difficultés de la modélisation du trafic et du réseau, cet objectif majeur pose de nombreuses problématiques. Comment calculer la gigue analytiquement pour un trafic modélisé par des distributions généralisées au niveau paquet ? Est-ce que les modèles proposés sont suffisamment simples et faciles à calculer ? Comment intégrer ces nouvelles formalisations pour le contrôle des performances ? Comment l'estimation analytique peut-elle minimiser le trafic des paquets de contrôle des connexions vidéo ?

Nous explorons tout d'abord le calcul de la gigue dans des files d'attente avec des trafics autres que le trafic Poisson. Ce dernier est largement utilisé pour modéliser le trafic sur Internet étant donnée sa simplicité en échange de la imprécision. L'idée pour le calcul de la gigue est d'utiliser, d'une part la même formule que le cas du Poisson mais en intégrant d'autres distributions, et d'autre part des approximations et des hypothèses quand la caractérisation analytique du temps de transit n'est pas possible. Nous adoptons la simulation pour valider les modèles approximatifs. L'ensemble de simulations montre que la gigue moyenne calculée par notre modèle et celle obtenue par simulation coïncident avec des intervalles de confiance adéquats. De plus, le temps de calcul estimé pour évaluer la gigue est minime, ce qui facilite l'utilisation des formules proposées dans des outils de contrôle et en optimisation.

Nous étudions par la suite la possibilité d'exploiter les résultats analytiques pour contrôler le tampon de gigue, un composant important dans la transmission vidéo. Nous estimons qu'il est possible d'évaluer ses performances analytiquement à travers l'estimation de la gigue dans ce type de tampon.

Nous proposons une approche algorithmique pour déterminer une modélisation stochastique en file d'attente dédiée à des connexions vidéo de bout-en-bout. Une validation expérimentale est choisie cette fois-ci. En plus de la vérification de l'hypothèse pour cette modélisation, nous fournissons une plateforme pour approximer la gigue réseau pour un flux vidéo.

Finalement, le dernier objectif est d'investir dans une gestion optimale du trafic de contrôle de la transmission vidéo. Pour réduire ce trafic de requêtes entre le client et le serveur, devenu de plus en plus intensif avec des applications telles que le *WebRTC* ou *Dynamic Adaptive streaming over HTTP*. Nous procédons à l'implémentation d'un algorithme de streaming adaptatif qui se base sur une mesure de la gigue du coté serveur. Ceci est possible grâce aux résultats obtenus dans les contributions précédentes.

# ABSTRACT

In recent years, we have witnessed the huge use of the Internet Protocol for delivering multimedia trafic. Developments in broadband networks led the progress in multimedia applications such as voice over IP, video streaming or real-time videos. However, the stochastic nature of the networks, in particular mobile networks, make it difficult to maintain a good quality at all times. The research of this PhD thesis deals with the improvement of the quality of service (QoS) for this kind of applications. Current network protocols provide multiple QoS control mechanism. Congestion control and transmission delay optimization are provided by packet scheduling strategies and bandwidth planning. Moreover, flow control adjusts the mismatch between the video server rate and the receiver available bandwidth. Nevertheless, video applications, in particular interactive videos, are very sensitive to delay variation, commonly called jitter. Indeed, the customers' perceived video quality depends on it. A jitter increase may cause a large video start-up delay, video interruptions and a decrease of image quality.

The main objective of this thesis is the study of jitter, which is not much studied in the IP literature. We also examine the impact of the increase of this parameter on video transmission. However, beyond the difficulties of modeling traffic and network, this major objective raises many other issues. How to calculate jitter analytically for traffic models with general distributions? Are the proposed models sufficiently simple and easy to calculate? How to integrate these new formalizations into performance monitoring? How can the analytical estimate minimize the traffic control packets exchange for each video connection?

We first explore the jitter calculation in queues with traffic other than Poisson traffic, that was widely used to model Internet traffic because of its simplicity. The idea is to compute jitter with the same formula for the Poisson traffic case, but with other distributions. For this, we need some approximations and assumptions when the analytical characterization of the transit time is not possible. We adopt simulations to validate the approximate models. The set of simulations shows that the average jitter calculated by our model and by simulation coincide within an appropriate confidence intervals. Moreover, the execution time to evaluate jitter is small, which facilitates the use of the proposed formulas in control tools and in optimization models.

We then study the possibility of exploiting this analytical results to control jitter buffers, an important component in the video transmission. We find that it is possible to evaluate its performances analytically by estimating jitter inside this type of buffer.

We propose an algorithmic approach to determine a queue model for end-to-end video connection. This time we chose an experimental validation. In addition to verifing the hypothesis for the proposed G/M/1 model, we provide a platform to approximate the network jitter for a video stream.

Finally, the last objective is to reduce control traffic of the video transmission that becomes increasingly intensive, especially in case of applications using protocols like *WebRTC* or *Dynamic Adaptive Streaming over HTTP*. We proceed to implement an adaptive streaming algorithm that is based on a jitter measurement on the server side. This is possible thanks to the results obtained in the previous contributions.

# TABLE DES MATIÈRES

# LISTE DES TABLEAUX

# LISTE DES FIGURES

# LISTE DES SIGLES ET ABRÉVIATIONS

| | |
|---|---|
| 3GPP | 3rd Generation Partnership Project |
| ACK | acknowledgment |
| ATM | Asynchronous Transfer Mode |
| AWAMP | One Way Active Measurement Protocol |
| CDN | Content Distribution Network |
| CBR | Constant Bit Rate |
| D | Deterministic |
| DASH | Dynamic Adaptice Streaming over HTTP |
| EF | Expedited Forwarding |
| FCFS | First-come, first-served |
| FIFO | First-in, First-out |
| G | General |
| GCC | Google Congestion Control |
| Gm | Gamma |
| GoP | Group of Picture |
| HD | High Definition |
| HQ | High Quality |
| MQ | Medium Quality |
| LQ | Low Quality |
| HTTP | Hypertext Transfer Protocol |
| ICMP | Internet Control Message Protocol |
| IETF | Internet Engineering Task Force |
| IOS | International Organization for Standardization |
| IPP | Intrrupted Poisson Process |
| IP | Internet Protocol |
| IPDV | Inter-packet delay variation |
| IPTV | TV over Internet Protocol |
| ISP | Internet Service Provider |
| ITU | International Telecommunication Union |
| LAN | Local Area Network |
| LogN | La log-normale |
| LTE | Long Term Evolution |
| M | Markovian |

| | |
|---|---|
| MMPP | Markov modulated poisson process |
| MOS | Mean opinion score |
| MPLS | Multiprotocol Label Switching |
| MPEG | Motion Picture Expert Group |
| MPPDV | Mean packet to packet delay variation |
| MTU | Message transmission unit |
| NS-2 | Network Simulator-2 |
| Nt | La loi Normale Tronquée |
| NTP | Network Time Protocol |
| PDF | Probablity Density Function |
| PDV | Packet delay variation |
| Pr | Pareto |
| PSNR | Peak-to-peak signal-to-noise ratio |
| QoS | Quality of Service |
| RTP | Real Time Protocol |
| RTCP | Real-time Transport Control Protocol |
| RTT | Round Trip Time |
| SLA | Service level agreement |
| SNMP | Simple Network Management Protocol |
| SVC | Scalable Video Coding |
| TCP | Transmission Control Protocol |
| TDM | Time-Division Multiplexing |
| UDP | User Datagram Protocol |
| VoIP | Voice over IP |
| VQM | Video quality metric |
| VNI | Visual Networking Index |
| WAN | WIde Area Network |
| WLAN | Wireless Local Area Network |

# CHAPITRE 1 INTRODUCTION

## 1.1 Contexte du projet

Le Protocole Internet (IP) est devenu de nos jours une partie fondamentale dans notre vie moderne. En observant l'évolution des systèmes de communication, on remarque le progrès des réseaux mobiles tel que le Long Term Evolution (LTE). Ils fournissent des réseaux à haut débit favorisant l'essor rapide des équipements de communication comme les téléphones intelligents et les tablettes. La diversité des plate-formes et des applications conduisent au succès des réseaux IP en augmentant le besoin d'être connecté à Internet chez les consommateurs. Le trafic IP dépassera même le seuil du *zettaoctet* ([ZO]=$10^{21}$ octet) à la fin de 2016 selon les rapports de *Cisco Visual Networking Index* (VNI) (Index, 2016).

Cette évolution du trafic Internet coïncide avec l'utilisation massive des applications vidéos. La figure 1.1 montre le trafic vidéo sur Internet comparé aux autres services. Il est bien clair que la vidéo est en tête de l'ensemble du trafic varié sur Internet avec un accroissement au cours des années de la vidéo interactive, dénotée par la « communication » vidéo sur l'histogramme. Il faut noter aussi le comportement des utilisateurs d'Internet dans les réseaux mobiles, où le trafic vidéo va compter pour 66% du trafic mobile total en 2020 (Index, 2016).

Diverses applications vidéo sont transportées par IP. Nous trouvons d'une part le *Streaming-Vidéo* par des services comme la *Vidéo-sur-Demande* et la *Télévision-sur-IP* (IPTV), d'autre part les vidéos interactives en temps réel offertes par des services comme la *Vidéo-Conférence* et la *Télé-médecine*. Ainsi, cet accroissement découles de la dominance des réseaux sociaux tels que *Facebook* et *Twitter* qui fournissent récemment une plateforme de vidéo streaming en direct via des services comme *Facebook LIVE* et *Periscope*. Cette croissance dépend aussi fortement des services vidéo comme Youtube, *Netflix* et *Akamai HD*, autant à cause de la diversité du contenu et la haute qualité (HD) des vidéos de divertissement que de l'avènement de l'ultra HD.

### 1.1.1 Transmission de la vidéo sur IP

Le fonctionnement d'une transmission vidéo à travers Internet suit généralement la chaine de transmission donnée dans la figure 1.2. Cette architecture est divisée en trois éléments principaux : l'émetteur, le réseau et le récepteur. Pour ce qui est de l'émetteur, on distingue deux types : pour une transmission de streaming vidéo, la source dispose de la vidéo stockée au préalable. Pour les communications interactives, la source capture la vidéo en temps réel,

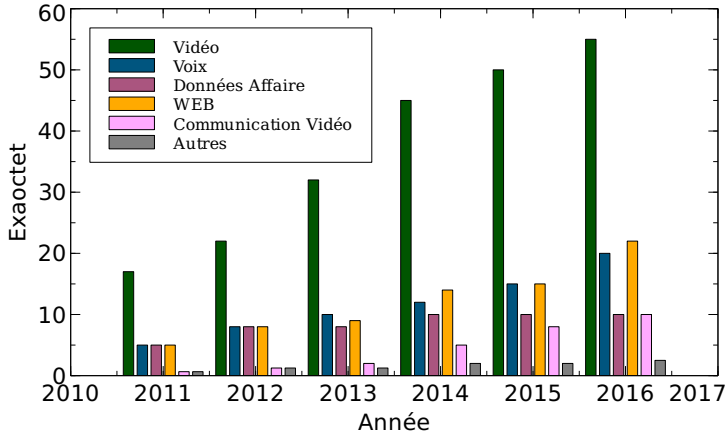



Figure 1.1 Estimation Mensuelle du Trafic Internet Global (Index, 2012)

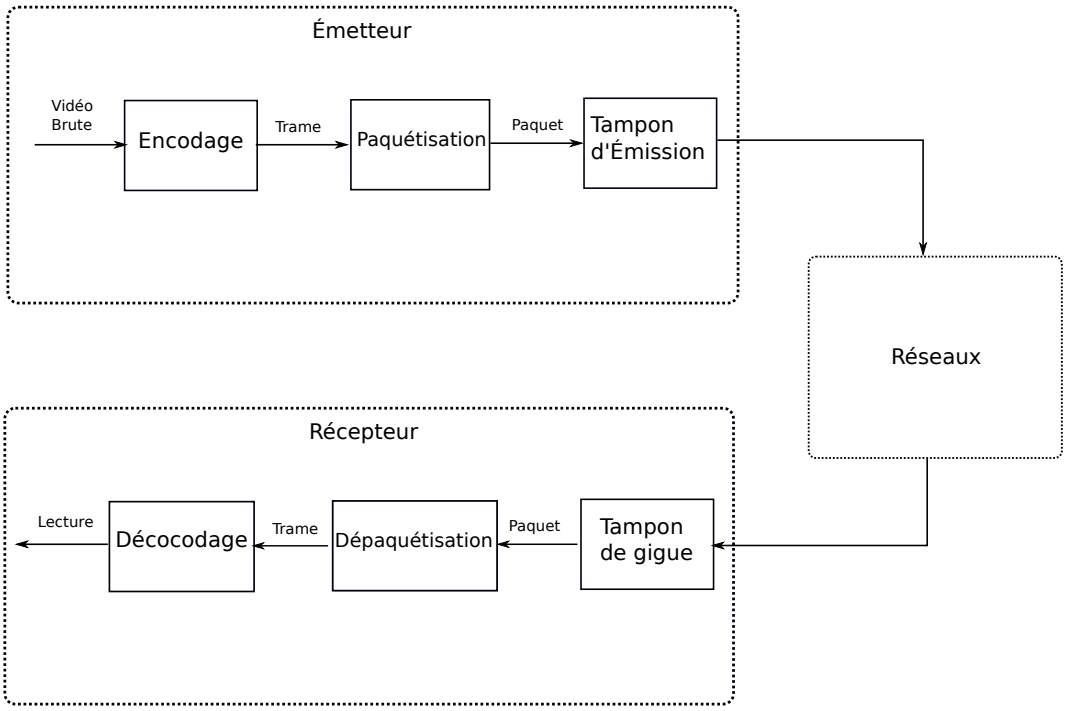



Figure 1.2 Architecture Générale des applications vidéo sur IP

par exemple par une caméra, pour qu'elle passe ensuite à l'encodeur. Quant au réseau, il comprend l'architecture physique, le routage et le protocole de transport utilisé. Enfin, le récepteur, qui représente la destination du trafic vidéo, est composé de 3 entités présentées dans la figure 1.2. Le récepteur n'est que le terminal utilisé pour afficher la vidéo. Il peut être connecté via les réseaux mobiles (3G ou LTE) ou les réseaux résidentiels (*Modem Câble*). Des notions introduites par la chaîne de transmission vidéo sont expliquées dans la suite.

**Encodage/décodage vidéo**

Un flux vidéo brut n'est qu'une séquence d'images fixes qui défilent dans une succession rapide pour donner l'illusion du mouvement. Les fichiers vidéo générés sont généralement de taille volumineuse (Hanzo et al., 2007).

L'encodage de la vidéo numérique brute est une étape très importante, puisque le transfert de tels fichiers sans compression pose un problème majeur pour les réseaux IP. Le concept principal derrière la compression vidéo est la suppression des informations redondantes à partir du signal d'origine pour réduire la quantité de données, en se basant ensuite sur la corrélation entre les images superposées pour reconstruire la vidéo initiale. Plusieurs algorithmes de compression sont proposés par les organismes de normalisation tels que l'International Organization for Standardization (IOS), l'International Telecommunication Union (ITU) ou le Motion Picture Expert Group (MPEG). Actuellement, les *codec*s les plus utilisés sont le MPEG-4 et le H.264. Il y a aussi un déploiement progressif du H.265 pour la résolution ultra HD.

La procédure du décodage effectuée par le lecteur de la vidéo est accomplie à l'aide des informations sur le profil d'encodage incorporées dans l'en-tête des trames générées par l'encodeur. Ce profil comprend les données sur les tailles des images en pixels, la période d'inter-images et la matrice de quantification. Cette unité reconstruit la vidéo initiale en recevant les trames après leur dépaquétisation avec un débit égal à celui de leur production par l'encodeur.

**Protocoles de transport**

Le transfert de la vidéo sur les réseaux Internet implique l'utilisation d'un protocole de couche transport. L'établissement de la connexion entre l'émetteur et le récepteur choisit le type de protocole en fonction du type de service et des exigences de qualité de service. Nous présentons ici les protocoles utilisés.

Le Transmission Control Protocol (TCP) est un protocole de la couche transport, normalisé par l'organisme de *Internet Engineering Task Force* (IETF) dans la RFC 793 (Braden, 1981).

Ce protocole est connu par sa fiabilité grâce à ses mécanismes de contrôle de flux et d'acquittement. Cependant, cette transmission contrôlée peut engendrer des délais supplémentaires et des congestions. Malgré cela, ce protocole gagne en popularité grâce à sa fiabilité et sa facilité de déploiement : il est de plus en plus utilisé grâce à l'émergence des streaming sur HTTP.

Le User Datagram Protocol (UDP) est l'un des principaux protocoles de transport, détaillé dans le rapport de l'IETF (RFC 768) (Postel, 1980). Ce protocole est pleinement utilisé pour les services temps-réel où la fiabilité n'est pas très importante comme pour les appels voix ou vidéo avec *Skype* ou *Google Talk*.

Le Real-Time Protocol (RTP) est un protocole de transport fonctionnant sur UDP qui est défini par l'IETF dans le RFC 3550 (Schulzrinne et al., 2003). Il est conçu pour transporter les services temps-réel tout en garantissant leur fiabilité en appliquant des nouveaux mécanismes pour le recouvrement des pertes de paquets et le contrôle de la congestion. Il est utilisé surtout par des applications de vidéoconférence proposées en version garantissant la QoS (orienté-QoS) par des compagnies comme *Cisco* ou *Tendberg* afin de fournir une plate-forme temps réel et fiable pour des services critiques comme la télé-médecine.

### 1.1.2 Exigence en qualité

Contrairement à la transmission des données, le trafic vidéo est caractérisé par sa sensibilité à la latence et sa tolérance à la perte des paquets. Les fournisseurs de service vidéo ont tendance à garantir une excellente transmission des données pour satisfaire leurs consommateurs. Ceci explique les exigences de plus en plus sévères en qualité de service. Néanmoins, IP se base essentiellement sur un comportement de meilleur effort (*Best Effort* en anglais) pour l'acheminement du trafic. Les réseaux IP sont donc caractérisés par leur nature stochastique, soit dans d'accès filaire, soit en sans-fil. Il est également basé sur le multiplexage statistique partout dans le réseau, ce qui ne répond pas à la sensibilité des applications vidéos et engendre en plus une dégradation de la qualité de service (QoS). La solution déployée pour résoudre ce problème est le contrôle de bout-en-bout en intégrant des mécanismes de contrôle de la transmission. Ceci est effectué par des mesures de QoS dont les principaux paramètres sont les suivants :

- Débit : désigne le volume de données en *bits* par unité de temps.
- Délai de transmission des paquets : est défini par le temps pris par un paquet entre la source et la destination à travers le réseau.
- Gigue : présente la variation des délais de transmission des paquets.
- Taux de perte : présente le taux de non délivrance de paquets d'un flux donné.

Les opérateurs et les fournisseurs de service vidéo tendent à minimiser le délai, la gigue et le taux de perte. Ils jugent aussi que de la vidéo, particulièrement en temps réel, est très sensible à la gigue. Une gigue élevée peut engendrer une distorsion du flux et une dégradation de la QoS. L'impact de la gigue sur la qualité perçue de la vidéo se résume en plusieurs facteurs gênants pour les consommateurs tels la pixelization de l'image, les interruptions de la vidéo appelées aussi problème de l'écran figé et le retard de la lecture de la vidéo (*start-up delay* en anglais). L'étude expérimentale de (Krishnan and Sitaraman, 2013) a montré que les utilisateurs sont impatients vis-à-vis ces facteurs dus à la gigue. Un tamponnage initial de plus de 2 secondes provoque l'abandon de la vidéo. En plus, un client qui subit des interruptions fréquentes de sa vidéo changera de fournisseur de service.

### Définition de la gigue

Les paquets transmis d'une source à une destination à travers les réseaux IP subissent plusieurs retards et arrivent à leur destination avec des délais variés. Cette variation du délai qui est appelée la gigue est due principalement à :

- la variation du temps d'attente à chaque nœud du réseau,
- la variation des chemins empruntés par les paquets pour atteindre leur destination. Cette variation a pour but d'éviter les nœuds congestionnés ou les liens en défaillance,
- la variation des délais d'ordonnancement,
- la perte de quelques paquets lorsque leurs retards dépassent le délai maximum autorisé.

De multiples définitions pour la gigue sont proposées par les organismes de standardisation. Nous trouvons, d'une part, *International Telecommunication Union-Telecommunication Standardization Sector* (ITU-T) qui définit la gigue sous deux formats :

La variation de délai par paquet (*PDV : Packet Delay Variation* en anglais) (itu, 2011) : la gigue $J$ ici est une mesure instantanée et elle est définie comme étant la différence entre le délai de transmission du $i^{ième}$ paquet $T_i$ d'un flux de paquets avec un délai référence $a_i$ qui peut être, soit le délai minimum, soit le délai moyen

$$J = T_i - a_i. \tag{1.1}$$

Cette définition a pris par la suite une autre forme (itu, 2006) : la différence entre le 99.9% centile du délai de transmission maximal $T_{max}$ et le délai de transmission minimal $T_{min}$ éprouvé par les paquets d'un flux vidéo dans une durée déterminée. Il est donné alors comme suit

$$J = T_{max} - T_{min}. \tag{1.2}$$

La moyenne de la variation de délai (*MAPDV : Mean Absolute Packet Delay Variation*) (Clark, 2003) : la gigue est donnée par la moyenne de l'équation (1.1), et cette différence est calculée en valeur absolue, ce qui donne

$$J = E[|T_i - a_i|]. \tag{1.3}$$

D'autre part, l'IETF propose :

Variation du délai inter-paquet (*IPDV : Inter-Packet Delay Variation*) : les RFCs 3393 et 3550 (Demichelis and Chimento, 2002; Schulzrinne et al., 2003) définissent la gigue comme une mesure instantanée définie comme la différence entre le délai de transmission de deux paquets consécutifs $i$ et $i + 1$ appartenant au même flux. Si $T_i$ et $T_{i+1}$ présentent ces deux délais, la gigue dans ce cas est

$$J = T_{i+1} - T_i. \tag{1.4}$$

Variation du délai moyen (*MPPDV : Mean Packet to Packet Delay Variation* ) : en RFC 4689 (Poretsky et al., 2006), IETF présente la gigue comme la moyenne de la valeur absolue de la mesure IPDV.

$$J = E[|T_{i+1} - T_i|]. \tag{1.5}$$

Les domaines d'application de ces définitions sont également variés. Dans ce contexte, l'étude effectuée par le RFC 5481 (Morton and Claise, 2009) clarifie l'applicabilité de ces mesures. Ils sont détaillés dans dans la section 1.1.3.

**Tampon de gigue**

Pour la majorité des applications, en particulier la vidéo, la variation dans le temps des arrivées des paquets doit être compensée pour avoir un flux régulier de paquets. Le décodeur doit relire les trames dépaquétisées avec le même taux d'émission que du coté de l'émetteur, d'où la nécessité d'un tampon de gigue (*Jitter Buffer* en anglais) présenté dans la figure 1.2. Il s'agit d'une mémoire au niveau du récepteur qui fonctionne en ajoutant un délai aux paquets pour enlever ou minimiser l'effet de la gigue.

L'ajustement du tampon de gigue joue alors un rôle critique dans la garantie d'une bonne qualité de transmission vidéo, particulièrement pour les services temps-réel. Dans ce cas, le temps passé en mémoire tampon ne doit pas dépasser un seuil donné. L'évaluation de la performance des tampons de gigue se basse essentiellement sur la capacité à éviter les pertes de paquets et les arrêts de lectures par le décodeur. Les pertes de paquets sont causées par la capacité finie de la file d'attente. Les arrêts de lectures sont du à une absence de paquets dans le tampon.

**Qualité d'expérience des utilisateurs (QoE)**

La qualité d'expérience est une mesure subjective qui évalue l'impression des utilisateurs d'un service. Il s'agit d'un indicateur relié à l'impact de la condition du réseau sur la qualité de service au niveau applicatif. Cette qualité est liée à la résolution de la vidéo, la facilité d'accès mais principalement à la fluidité de la lecture de la vidéo. Le dernier critère est associé à la performance du tampon de gigue. Cette mesure intéresse beaucoup les fournisseurs de service puisqu'il s'agit d'un indicateur de la satisfaction de leur clientèle. Malgré la corrélation implicite entre la QoS et la QoE, la mesure de cette dernière n'implique pas nécessairement des mesures de la QoS. Elle est une métrique mesurée au niveau de la couche application. Ceci mène sa popularité chez les fournisseurs vidéo. Elle facilite le contrôle de leurs services sans avoir besoins des mesures de la QoS auxquelles ils n'ont généralement pas accès.

Plusieurs méthodes sont proposées pour la mesure de la QoE. Les plus importantes sont la notation de la vidéo par les observateurs. Ce score est appelé *Mean Opinion Score* (MOS) et il varie généralement entre 1 et 5. La valeur 1 signifie une mauvaise qualité et 5 une excellente qualité. Cette méthode est utilisée fréquemment par *Skype* et *Facebook*. Toutefois, parce que cette technique est coûteuse en termes de temps et d'argent, les chercheurs ont proposé une approximation de la QoE objectivement à travers des paramètres liés à la qualité de la vidéo comme *video quality metrics* (VQM) et *peak-to-peak signal-to-noise ratio* (PSNR) ou bien une façon plus simplifiée. D'autres approches proposent d'utiliser le statut du tampon à gigue pour l'évaluation analytique de la QoE.

### 1.1.3 Domaine d'applicabilité de la gigue

Diverses tâches reliées à la transmission multimédia font appel à la mesure de la gigue. Tout d'abord, la détermination de la taille du tampon de gigue, mesurée soit en délai, soit en nombre d'octets. Les concepteurs des applications ont besoin de connaître une approximation de la gigue dont ils doivent tenir compte pour concevoir un tampon optimal et adaptatif. Ce design est évalué par la suite par rapport à la QoE.

La gigue, en outre, permet de déduire l'occupation des files d'attente sur le chemin emprunté par les paquets et elle est utilisée comme un indicateur de congestion. En effet, son usage est plus fréquent par rapport aux délai de transmission, puisque sa mesure n'exige pas la synchronisation entre les machines source et destination.

La gigue est souvent intégrée comme une métrique dans les accords de niveau de service (*Service-Level Agreement (SLA)* en anglais) dans des réseaux dédiés aux services à haute

qualité comme ceux utilisant Multiprotocol Label Switching (MPLS). Dans ce cas, la minimisation de la gigue peut alors influencer le routage du trafic.

Récemment, l'estimation de la gigue est devenue aussi indispensable pour la modélisation de la QoE d'un service multimédia et particulièrement les services vidéo. Ceci est non seulement dû à la corrélation entre la QoS et la QoE mais aussi à l'évaluation des performances des tampons de gigue qui implique indirectement la gigue.

## 1.2 Éléments de problématique

Les systèmes de télécommunication actuels déterminent la gigue dans le réseau en procédant par une estimation active utilisant des sondes de mesure qui génèrent un flux dédié pour le contrôle et rapportent les mesures à un serveur central. Une autre méthode de mesure active est effectuée sur un échantillon de paquet. Les informations nécessaires pour faire cette évaluation, comme les temps d'envoi et de réception de chaque paquet sont collectées à partir des *Timestamps* pour les flux RTP ou à partir des paquets d'acquittement (Ack) pour les flux TCP. Quant aux services qui utilisent UDP, on ne peut pas la mesurer. Cette technique peut être appliquée à un contrôle temps réel mais il est difficile de l'intégrer dans la modélisation et la conception des réseaux ou de l'utiliser dans des programmes d'optimisation, tels que le dimensionnement des tampons à gigue, pour les algorithmes de routage, ou dans l'allocation des ressources. Une modélisation analytique de la gigue devient donc une nécessité. Évidemment, cette modélisation doit être à la fois rapide et suffisamment précise pour répondre à la sensibilité des applications vidéo.

### 1.2.1 Modélisation analytique de la gigue

La gigue intervient de plus en plus dans des domaines qui nécessitent une modélisation analytique. En effet, à coté de la minimisation des coûts, l'utilisation de la bande passante et le délai, la minimisation de la gigue est exigée pour toute application orientée vers la qualité de service.

Nous citons en premier lieu quelques exemples de ces domaines :

1. De nos jours, un grand nombre de chercheurs en transmission vidéo sur Internet et de concepteurs réseau qui s'intéressent à un transfert orienté-QoE produisent des algorithmes basés sur l'estimation de la QoE. Des exemples comme l'allocation des ressources radio en LTE, ou l'adaptation dynamique de la qualité de la vidéo avec la bande passante, gagnent de la popularité chez les fournisseurs de service vu leur fiabilité, leur simplicité et la possibilité de mesurer la QoE au niveau applicatif. Ces

algorithmes, par contre, nécessitent la modélisation analytique de la QoE, la gigue en premier lieu, compte tenu de son impact sur la QoE.

2. Les réseaux MPLS exigent un routage orienté QoS. Dans ce cas, il faut définir un algorithme de sélection du chemin utilisé par les paquets d'un flux en se basant sur des exigences en QoS (débit, délai et gigue). Afin de définir cet algorithme de routage qui prend en considération la gigue comme une contrainte de transmission, il faut une expression simplifiée pour la gigue.
3. L'estimation de la gigue joue aussi un rôle crucial pour éviter les pertes dans les tampons de gigue. Il faut trouver un compromis entre les pertes et le délai passés dans le tampon afin d'éviter :
   — le débordement de la file d'attente en cas de rafale pour éviter la perte des paquets,
   — l'absence de paquets dans la file d'attente. Dans ce cas, le problème est différent. Si le tampon se vide, le flot de paquets vers l'application va s'arrêter. Pour un service vidéo, l'image va se figer jusqu'à la reprise du flux de paquets..

   Le dimensionnement du tampon de gigue et l'évaluation de ses performances rend obligatoire l'estimation analytique de la gigue.
4. Un autre domaine en plein émergence est l'allocation de ressources virtuelles et l'emplacement des bases de données dans le réseau *Infonuagique* (*Cloud Computing* en anglais). Ces fonctionnalités sont sensibles à la QoS et à la gigue en particulier et ils sont basés sur une approche d'optimisation en nombre entiers. Une formulation analytique de la gigue est donc requise.

Tous ces domaines d'application montrent l'importance de fournir des expressions mathématique, soit pour la gigue de bout-en-bout, soit sur un seul lien, comme est le cas des tampons de gigue. Cela implique en premier lieu la modélisation des réseaux et la caractérisation du trafic à l'aide des systèmes de files d'attente. Souvent, le modèle le plus utilisé pour le processus d'arrivée des paquets dans le réseau Internet est le modèle de Poisson. Jusqu'à maintenant la plupart des travaux d'études de performance dans ce réseau se basent essentiellement sur la considération des nœuds du réseau comme des files d'attente de type M/M/1. L'avancement de la recherche dans le domaine de la caractérisation du trafic qui circule sur IP montre que le processus de Poisson n'est valide que dans le niveau session, et il n'est pas adéquat pour modéliser le processus d'inter-arrivée au niveau de la couche réseau. Cela est essentiellement le résultat de la dépendance entre les paquets générés, surtout en cas de la vidéo, due à l'effet de la compression. En plus, la taille des paquets qui circulent sur IP ont généralement une distribution tri-modale (Fraleigh et al., 2003) qui correspond à la combinaison des paquets du rapport de contrôle du TCP ou RTP et l'existence des deux *Message Transmission Unit* (MTU) ordinaires de tailles respective 572 et 1500 octets dans le réseau dorsal de l'Internet.

Il y a donc une tendance récente d'abandonner l'utilisation des M/M/1, surtout dans les réseaux mobiles ou pour les tampons de gigue. La planification des réseaux emploie des modèles plus avancés pour le trafic sur IP comme le processus *Markov Modulated Poisson Process* à deux états (MMPP-2) (Muscariello et al., 2004) ou par un *Interrupted Poisson Process* (IPP) (Dahmouni et al., 2005). Des modèles comme M/D/1 et G/G/1 sont aussi utilisés pour les tampons de gigue pour s'approcher du trafic réel. Cependant, ce progrès reste toujours limité et ce, parce que deux problèmes majeurs se manifestent : d'une part, les modèles généralisés ne sont exploités que pour l'estimation du taux de perte ou du délai de transmission et pour l'allocation de la capacité, sans donner de l'importance à la gigue, à cause de la complexité de la tâche. D'autre part, les modèles qui offrent des méthode pour estimer analytiquement la gigue sont difficilement utilisables, soit à cause de la complexité de calcul, soit par leur imprécision pour modéliser le trafic vidéo. Cela rend difficile leur intégration dans des domaines qui nécessitent la précision ou beaucoup de calculs.

Le défi est alors non seulement de fournir une formulation analytique à la gigue mais aussi de faire en sorte que le modèle proposé soit simple et facile à calculer pour faciliter son intégration dans les outils de conception et dans les programmes d'optimisation. Par ailleurs, étant donné l'imprécision du modèle de Poisson, il faut développer une technique qui modélise la gigue pour un trafic non-Poissonien.

### 1.2.2 Quelle définition de la gigue faut-il utiliser ?

La multitude des définitions de gigue engendre un autre problème lié à la pertinence de chaque définition pour un service donné. Une comparaison entre ces définitions est présentée dans le RFC 5481. La distribution du PDV donné par l'équation (1.1) distingue les variations du délai à long terme et prend la même forme que la distribution du délai de transmission. Elle reflète le temps d'attente additionnel dans les nœuds du réseau. Cette définition, avec celle de l'équation (1.2), sont très adéquates pour le dimensionnement des tampons de gigue. Par contre, la définition de l'IETF donnée par l'équation (1.4) est une bonne mesure de la capacité du réseau à préserver l'espacement entre les paquets. Elle est utilisée dans les cas où la variation à court terme est importante, comme pour la détection de la congestion. La mesure de la moyenne donnée par l'équation (1.5) est largement utilisée pour les mesures actives. En pratique, certains préfèrent cette dernière définition malgré qu'elle sacrifie le concept de deux facettes de la gigue, les valeurs positives et négatives, puisque, d'une part, elle est la plus simple, et d'autre part, la seule définition qui possède une modélisation analytique simplifiée.

De plus, l'étude du RFC 5481 a accentué le fait qu'il n'existe aucune garantie que la gigue en tant que MPPDV est une bonne mesure pour modéliser la QoE, ainsi que pour évaluer ou contrôler les tampons de gigue.

### 1.2.3 Modélisation de réseaux

De même que l'estimation analytique de la gigue, comprendre le comportement du trafic qui passe par les réseaux IP et la caractérisation de ce trafic surtout pour le cas de la vidéo est une nécessité et elle fait généralement appel à la modélisation en files d'attente. Les considérations traditionnelles des files d'attente sont, soit compliquées, à cause de la modélisation des chemins empruntés par les paquets en une série de files d'attente, soit imprécises, à cause de l'utilisation du modèle de Poisson pour gagner en simplicité. Bien que la planification moderne des réseaux IP a besoin d'une modélisation qui sert à simplifier les mesures de la QoS. En plus, il doit prendre en considération la généralité du trafic et limite les échanges des messages de contrôle de la connexion.

### 1.2.4 Trafic de contrôle optimal pour les transmissions vidéo

Actuellement, dans la plupart des applications vidéo où il faut estimer la qualité de service (gigue, bande-passante ou perte), comme en cas du streaming adaptatif, le contrôle de congestion, ou en MPLS, les paramètres de la QoS sont calculés par le récepteur pour qu'ils soient par la suite transférés à l'émetteur. Les fournisseurs de service se servent beaucoup de cette solution vu sa simplicité ; elle n'exige pas la connaissance du réseau, le tout se fait aux deux bouts de la transmission. La technique pour estimer la gigue se résume par le marquage de chaque paquet $i$ avec le temps d'envoi $t_e(i)$ à son émission et puis avec le temps de réception $t_r(i)$ lors de son réception. La gigue instantanée est calculée pour ce protocole par l'équation suivante

$$J = |(t_r(i+1) - t_r(i)) - (t_e(i+1) - t_e(i))|. \tag{1.6}$$

Une moyenne de cette mesure est calculée sur un échantillon de paquets. Bien que, cette méthode ne soit utilisée qu'avec RTP, le TCP par contre n'exploite pas ces informations sur $t_e(i)$ et $t_r(i)$ pour ses calculs de la gigue ou pour estimer la QoE. Dans le cas du streaming vidéo, l'émetteur est un serveur et il est chargé de contrôler la connexion établie pour la transmission vidéo de bout-en-bout créée dans ce cas avec TCP. Le serveur, qui est responsable de la gestion de plusieurs sessions en même temps, reçoit régulièrement les Acks (*packet acknowledge*) des récepteurs qui contiennent plusieurs informations à propos de la connexion. Ils sont lus par le serveur pour recouvrer les erreurs et pour contrôler la congestion. De plus,

dans les protocoles de streaming récents, le serveur reçoit aussi des rapports supplémentaires sur l'état de la lecture de la vidéo. Par conséquence, ce trafic de contrôle devient très intense pour la TCP et le RTP et ceci peut engendrer la congestion et une latence en temps de réponse pour le contrôle.

Une façon de gestion optimale des informations de contrôle est de bien exploiter les informations qui circulent déjà sur le réseau sans ajouter d'autres trafics de contrôle pour l'ajustement de la transmission avec la condition du réseau.

## 1.3 Contribution

Après la mise en contexte du projet et après avoir clarifié les problèmes à traiter, il est possible maintenant d'identifier notre contribution. Quatre idées maîtresses sous-tendent nos objectifs de recherche : analyser la gigue analytiquement pour des processus généralisés, modéliser une connexion vidéo de façon simple et précise, utiliser la gigue pour évaluer la performance des tampons de gigue et finalement, contrôler des applications vidéos afin d'améliorer la QoE des utilisateurs et de bien gérer les flux de contrôle. Cela a produit les contributions suivantes :

**Calcul de la gigue** Nous calculons la gigue pour un trafic non-Poisson analytiquement en une seule file d'attente infinie avec le principe du premier arrivé premier servi (FCFS) et avec un seul flux. Nous fournissons pour la première fois des expressions simplifiées et rapides pour calculer la gigue dans différents types de files d'attente qui modélisent les nœuds du réseau. En effet, l'estimation de la gigue à cette étape présente le départ de l'étude de la gigue d'une façon analytique et simplifiée. Cette analyse repose principalement sur la définition de la gigue donnée par MPPDV. Cette étude présente une clé non seulement pour appliquer la gigue dans des domaines qui nécessitent cette formulation analytique, mais aussi pour estimer par la suite la gigue dans le réseau.

Nous allons aussi étudier la possibilité de calculer la gigue dans des files d'attente à capacité finie, qui est généralement le cas dans les nœuds du réseau d'accès et, en particulier, des tampons de gigue. Nous avons démontré que dans certains cas, on peut utiliser le modèle valable pour des files d'attente infinies en cas des files d'attente finies.

**Modélisation de réseau** Nous modélisons le réseau de bout-en-bout pour des connexions vidéo par un système de files d'attentes précis et simple afin que l'estimation des paramètres de QoS soit rapide à évaluer avec des méthodes analytiques. Cette étude utilise la gigue pour vérifier l'exactitude du modèle proposé. Cela va produire une méthode analytique pour estimer la gigue pour un trafic vidéo en plus de la caractérisation de la file d'attente utilisée

pour la modélisation. Une connexion vidéo peut être modélisée par une file d'attente G/M/1 et la caractérisation de ses paramètres est possible à travers des méthodes analytiques ou numériques.

**La gigue MPPDV pour le contrôle des tampons de gigue** Nous étudions la relation entre la gigue définie par le MPPDV en une seule file d'attente et la performance des tampons de gigue. Cette analyse de corrélation mène à un algorithme d'estimation analytique des principales métriques des tampons de gigue, ce qui va servir à modéliser analytiquement la QoE et à aider au dimensionnement de la mémoire de ces tampons.

**Gestion du trafic du contrôle** Nous proposons un algorithme côté serveur d'adaptation dynamique de la qualité de la vidéo sur TCP basé sur l'estimation analytique de la gigue comme indicateur de l'état du tampon de gigue. Ceci présentera une nouvelle vision pour les algorithmes d'adaptation du niveau de la qualité en streaming qui gère les informations qui circulent sur le réseau d'une façon optimale.

# CHAPITRE 2 REVUE CRITIQUE DE LA LITTÉRATURE

Ce chapitre comprend une revue de littérature reliée au travail de recherche de cette thèse. La répartition de ce chapitre se fait en trois parties qui couvrent les différents axes de recherche. Nous récapitulons les recherches sur le calcul de la gigue et les modèle des réseaux utilisés, les processus proposés pour le contrôle des tampons de gigue et les algorithmes d'adaptation dynamique de la transmission vidéo.

## 2.1 Calcul de la gigue

### 2.1.1 ATM

De nombreuses études ont été consacrées à l'estimation de la gigue dans les réseaux *Asynchronous Transfer Mode* (ATM) pendant les années 90. L'ATM est un réseau à commutation de paquets et la transmission se fait en mode connecté. Les paquets générés par les différentes sources sont découpés en « cellules » de taille fixe de 53 octets multiplexés avec d'autres flux à chaque nœud. Le succès des réseaux ATM est fortement lié à leur capacité de fournir la qualité de service nécessaire pour chacune des classes de trafic prises en charge par le réseau. Principalement, ce type de réseau vise des valeurs très faibles de taux de perte de l'ordre de $10^{-8}$. Dans (Roberts and Guillemin, 1992), la gigue est considérée comme un processus à temps discret et la corrélation entre les délais de transmission successifs des cellules est supposée Markovienne caractérisée par la matrice de transition $Q$. Cette recherche propose une distribution de la gigue pour un trafic périodique et l'auteur suppose que cette analyse est efficace pour le dimensionnement des techniques de conformité du trafic comme le seau percé (*Leacky Bucket* en anglais) dans certains cas.

Comme le trafic *Constant Bit Rate* (CBR), qui génère périodiquement des cellules ATM, représente une portion significative du trafic global sous ATM, il existe une multitude de travaux s'intéressant au calcul de la gigue pour ce type de flux. Une caractérisation complète de la gigue dans le cas d'un seul nœud et dans le cas de plusieurs nœuds est donnée par (Matragi et al., 1994, 1997). Dans un premier temps, l'auteur a déterminé la distribution stationnaire de la gigue pour un flux marqué, périodique et multiplexé par un trafic de fond aléatoire et corrélé. La transformée en $Z$ de la gigue dans un seul nœud ATM est donnée par

$$J(z) = zG(\frac{B(z)}{z}) + \sum_{m=1}^{G_{max}} g(m) \left(\frac{B(z)}{z}\right)^{m-1} (1 - z^{-1})C_m(z), \tag{2.1}$$

$$C_m(z) = \sum_{k=1}^{m-1} (\frac{B(z)}{z})^k \sum_{i=0}^{k-1} z^{-i} \pi_k(0,i) Pr[Q_1 = i] \tag{2.2}$$

où $Q_1$ est la variable aléatoire dénotant le nombre de cellules dans le système et $\pi_k(0,i)$ dénote la probabilité que le système soit vide dans le $k^{\text{e}}$ intervalle de temps, $g$ représente la densité de probabilité de la variable aléatoire du temps d'inter-arrivée, $G(z)$ sa transformée en $Z$ et $B(z)$ représente la fonction génératrice du trafic de fond.

La deuxième partie du travail concerne le calcul de la gigue des cellules ATM traversant plusieurs nœuds. L'estimation de la gigue de bout-en-bout d'un flux périodique se base sur une approche approximative. Les résultats pour un trafic $\rho \approx 1$ donnent une valeur

$$J(z) = zG(B(z)). \tag{2.3}$$

Dans le cas où $\rho \approx 0$ , le calcul de la gigue du flux marqué nécessite le développement en série de Taylor au premier ordre de la fonction génératrice des paquets de background $B(z)$ ainsi que celle de la longueur de la file $Q(z)$

$$B(z,\rho) = 1 - \rho + \rho a(z) + O(\rho^2) \tag{2.4}$$

$$Q(z,\rho) = 1 + \rho(\frac{a(z) - z}{z - 1}) \ . \tag{2.5}$$

A partir des équations (2.4) et (2.5), le développement en série au premier ordre de la fonction $J(z)$ devient

$$J(z) = G(z) \left[ 1 + \frac{\rho}{2} \left( \frac{a(z) - 1}{z - 1} - 1 \right) \right]. \tag{2.6}$$

Une analyse similaire de la gigue pour un trafic périodique (Privalov and Sohraby, 1998) s'intéresse à un flux individuel en considérant que même le trafic de fond est périodique. L'auteur suggère une méthode de calcul de la gigue dans un multiplexeur homogène et une autre pour le cas hétérogène, en donnant la distribution de la gigue d'un flux à partir de la probabilité jointe de tous les trafics.

Les différents modèles pour le calcul de la gigue en ATM ne sont pas applicables pour les réseaux IP. En effet, en ATM, les paquets sont de petite taille et de longueur constante, et les sources sont généralement périodiques. En plus, la seule mesure de QoS considérée est la probabilité de perte de cellules, qui est très faible, de sorte que les résultats sont basés sur la limite de Chernoff. Ceci est très différent des réseaux IP, où les paquets ont des longueurs différentes, les processus d'arrivée sont loin d'être périodiques et les exigences en QoS sont basées sur le délai, la perte, la gigue et la bande passante.

### 2.1.2 Internet Protocol

Internet est devenu un réseau avec des mécanismes de QoS. Après l'apparition du MPLS, il y a eu une migration vers ce réseau, et par conséquent, beaucoup de recherches ont commencé à s'intéresser à la gigue en IP. Les chercheurs ont essayé de trouver des résultats généraux pour la gigue. Dans (Brun et al., 2006), une approximation analytique est proposée pour la gigue de bout-en-bout d'un trafic périodique avec un trafic de background, tous les deux régis par un processus de renouvellement. La gigue, dans ce cas, est donnée par la moyenne de la valeur absolue de la somme de la variation $\Delta_k$ de l'intervalle entre les paquets au nœud $k$

$$J_{[1..n]}(T) = E\left[\left|\sum_{k=1}^{n} \Delta_k\right|\right], \tag{2.7}$$

$$\Delta_k = W_1(k) - W_0(k), \tag{2.8}$$

A partir de cette formule, il est possible de trouver une expression pour la gigue en trouvant une approximation de la fonction de densité de probabilité de la variable aléatoire $\sum_{k=1}^{n} \Delta_k$. Des études récentes très approfondies proposent des modèles pour la gigue pour des réseaux à différenciation de services (*DiffServ*). Les auteurs de (Alshaer and Elmirghani, 2008) définissent la fonction génératrice de la gigue pour un trafic On-Off de classe *expedited forwarding* (EF) et ils mettent en évidence l'importance de cette étude pour le trafic temps-réel.

En effet, toutes ces méthodes se concentrent sur l'analyse de la gigue en donnant, soit des méthodes probabilistes, ce qui permet difficilement d'obtenir des expressions simples, soit des expressions compliquées, qui engendrent un temps de calcul très grand.

En réalité, le contrôle de la gigue fait par les opérateurs et les fournisseurs de service dans les réseaux qui assurent le transport des applications vidéo est basé essentiellement sur des mesures effectuées sur des liens de transmission. Ils disposent d'un serveur et de logiciels dédiés à la synchronisation et l'envoi des paquets Internet Control Message Protocol (ICMP) *echo* ou avec *Simple Network Management Protocol* (SNMP) pour recueillir des informations afin d'estimer la gigue (Cisco, 2007). Cette méthode n'est pas assez précise puisque les paquets ICMP ou SNMP sont souvent traités par les routeurs comme un trafic non-prioritaire. Par ailleurs, des outils plus modernes comme le *One-Way Active Measurement Protocol* (OWAMP) sont conçus pour mesurer les délais de transmission (Skurowski et al., 2010). L'idée de ce protocole est d'envoyer des paquets qui contiennent un estampille temporelle pour synchroniser les horloges. Ensuite le récepteur marque le paquet par le temps d'arrivée de ce paquet. Malgré que la prédiction active des paramètres de la qualité de ser-

vice soit très utile, elle engendre d'autres problèmes, comme l'augmentation du trafic par les paquets de contrôle et l'ajout de délais supplémentaires.

Une étude récente (Dahmouni et al., 2012a), propose un modèle simple et rapide à calculer. Ce travail produit des expressions dont certaines sont exactes et d'autres approximatives pour estimer la gigue dans un seul nœud pour un processus d'arrivée Poisson, ainsi que pour un chemin de files d'attente M/M/1. Bien que le modèle ne soit pas toujours précis dans le cas des réseaux IP, la simplicité des expressions rend le travail très utile pour la gestion des réseaux. Ce travail en particulier, a une grande utilité pour notre méthodologie de recherche. Nous allons nous inspirer des approximations et des formules données pour les généraliser dans le cas non-Poisson.

Le modèle proposé par (Dahmouni et al., 2012a) est valide pour des files d'attente de type M/M/1. Le temps d'inter-arrivée est donc exponentiel et les paquets reçoivent chacun à leur tour, un service d'une certaine durée qui suit une loi exponentielle de paramètre $\mu > 0$. Chaque type de flux à l'entrée de ces files d'attente a un taux d'arrivée défini comme suit :

$\lambda$ le taux d'arrivée total pour $K$ flux $\lambda = \sum_{j=1}^{K} \lambda_j$

$\lambda_k$ le taux d'arrivée du flux marqué

$\lambda_0$ le taux d'arrivée du flux de background ($\lambda_0 = \lambda - \lambda_k$).

Il faut tout d'abord définir les variables pour un paquet $i$ :

$t_i$ temps d'arrivée, et $R_i = t_i - t_{i+1}$ est le temps d'inter-arrivée.

$r_i$ temps de départ.

$W_i$ temps d'attente.

$S_i$ temps de service.

$T_i$ temps de transit avec $T_i = W_i + S_i$.

### Cas d'un seul nœud M/M/1

Dans le but de trouver une formule simple pour le cas d'un seul nœud, les auteurs de (Dahmouni et al., 2012a) ont fait un ensemble d'approximations en visant plusieurs cas.

**Taux d'arrivée faible** La première étape du travail était de considérer un flux marqué par un taux d'arrivée faible ($\lambda_k << \lambda$). Dans ce cas, deux paquets consécutifs du flux $k$ sont séparés par un grand nombre de paquets des autres flux. Cela signifie qu'on peut ignorer la corrélation entre les délais de transit de ces deux paquets. Cette approximation nous amène à dire que si $T_j$ et $T_{j+1}$ sont deux variables aléatoires du temps de transit des paquets $j$ et $j+1$, alors on peut les considérer comme indépendantes.

On sait par ailleurs que pour deux variables aléatoires $X_1$ et $X_2$ continues, indépendantes et de même distribution $f$, la variable $\mid X_1 - X_2 \mid$ suit la distribution $f$ si et seulement si $f$ est une distribution exponentielle. Puisque $T_i$ et $T_{i+1}$ sont deux variables aléatoires indépendantes, continues et possèdent la même distribution exponentielle de paramètre $\eta = \mu - \lambda$ où $\mu$ et $\lambda$ sont respectivement le taux de service et le taux d'arrivée, alors, $\mid T_{i+1} - T_i \mid$ suit la loi exponentielle de paramètre $\eta$. Par la suite, la gigue $J$ peut être facilement calculée en prenant la moyenne de la différence entre les délais de transit des paquets, soit

$$J = E[\mid T_{j+1} - T_j \mid] = \frac{1}{\eta}. \tag{2.9}$$

**Taux d'arrivée élevé** La deuxième approximation est de considérer un taux d'arrivée élevé pour les paquets du flux marqué ($\lambda_k \approx \lambda$ ). De ce fait, nous pouvons négliger la présence du flux de *background*.

Si le paquet $n+1$ arrive à la file d'attente et trouve que le paquet $n$ est déjà traité ($r_n \leq t_{n+1}$), alors le temps d'attente $W_{n+1} = 0$. Sinon, le temps d'attente est égal au temps de transit du paquet $n$ moins le temps d'inter-arrivée. On peut écrire

$$\begin{aligned} J &= E[\mid T_{i+1} - T_i \mid] \\ &= \int_0^\infty R(y) \left\{ \int_0^\infty S(s) \left[ \int_0^i \mid s - x \mid T(x)dx + \mid s - i \mid \int_i^\infty T(x)dx \right] ds \right\} dy. \end{aligned} \tag{2.10}$$

Pour une file d'attente M/M/1, $T(x) = \eta e^{-\eta x}$, $S(s) = \mu e^{-\mu s}$ et $R(y) = \lambda e^{-\lambda y}$. La simplification de la formule (2.10) donne :

$$J = E[\mid T_{i+1} - T_i \mid] = \frac{1}{\mu} \tag{2.11}$$

**Taux d'arrivée intermédiaire** Les auteurs de (Dahmouni et al., 2012a) ont prévu aussi le cas d'un flux marqué avec un taux d'arrivée moyen. Ce flux est multiplexé par un flux de background Poisson de paramètre $\lambda_0$. La définition de la gigue pour ce flux devient

$$J^k = E[\mid T_i^k - T_{i-1}^k \mid]. \tag{2.12}$$

Ils proposent l'approximation

$$J^k \approx \frac{1}{\eta} f(\tau_k, \eta) \tag{2.13}$$

$$f(\tau_k, \eta) = 1 - e^{-\eta\tau_k}(\eta\tau_k + e^{-\eta\tau_k}) \tag{2.14}$$

$$\tau_k = \frac{1}{\lambda_k}. \tag{2.15}$$

L'expression $1-f(\tau_k, \eta)$ est l'estimation du degré de corrélation entre deux paquets consécutifs du flux k.

### Cas d'un chemin de files d'attente M/M/1

L'estimation de la gigue à travers un chemin de files d'attente est plus complexe que le cas d'un seul nœud. Il faut prendre en considération la corrélation entre le temps de service dans le nœud et le temps d'arrivée au nœud suivant. La définition de la gigue pour un flux qui passe via $N$ files en série est donnée par

$$J_{[1..N]} = E\left[\left|\sum_{n=1}^{N} T_{j+1}(n) - T_j(n)\right|\right], \tag{2.16}$$

avec $T_i(n)$ le temps de transit du paquet $i$ au nœud $n$.

L'effet de la dépendance entre le délai de transit dans un nœud et le temps de service dans le nœud précédent se manifeste par la diminution de la gigue d'un nœud à un autre, d'où la déduction de la formule analytique qui suit

$$J^k_{[1..N]} = \frac{1}{\eta_1} f(\tau_k, \eta_1) + \sum_{n=2}^{N} \frac{1}{\eta_n} K(\lambda_k^{(n)}, \eta_n) f(\tau_k^{(n)}, \eta_n), \tag{2.17}$$

$$\tau_k^{(n)} = \tau_k + \sum_{i=2}^{n} J^k_{i-1} \tag{2.18}$$

où $\eta_n$ est le taux de séjour au nœud $n$ et $K(\lambda_k^{(n)})$ présente la corrélation entre les délais des deux paquets consécutifs, il est possible de l'écrire en fonction de la taille de la file d'attente $n$, $q_i^{(n)}$

$$K(\lambda_k^{(n)}, \eta_n) = 1 - E\left[q_i^{(n)} q_{i+1}^{(n)}\right] \tag{2.19}$$

$$= 1 - \int_0^\infty R_q^{(n)}(t) g^{(n)}(t) dt, \tag{2.20}$$

où $g^{(n)}(t)$ et $R_q^{(n)}(t)$ représentent la distribution du temps d'inter-arrivée et la fonction d'auto-corrélation des $q_i^{(n)}$.

### 2.1.3 Échec du modèle de Poisson

Au cours des dernières années, les chercheurs ont mis beaucoup d'efforts pour comprendre la nature du trafic Internet sont élaborés. Il y a eu de nombreuses recherches pour la caractérisation des inter-arrivées et la taille des paquets dans les nœuds d'un réseau. Les propriétés statistiques trouvées pour cette modélisation se résument dans l'auto-similarité, la distribution à queue lourde et la convergence vers un processus de Poisson ou un trafic MMPP.

Le modèle décrit dans les sections (2.1.2) et (2.2) suppose que le réseau IP est composé de files d'attente M/M/1. Cependant, cela n'est exact que dans un certain nombre de cas, comme pour un processus de trafic hiérarchique MMPP ou dans des nœuds où il y a agrégation de plusieurs trafics HTTP. De nombreux travaux (Paxson and Floyd, 1995; Hayman, 2005; Leland et al., 1994; Liang et al., 2011) montrent que la distribution d'inter-arrivée des paquets dans les réseaux locaux ou étendus est différente de la distribution exponentielle et donc le processus n'est pas Poisson. De plus, la taille des paquets des applications multimédia au niveau local ou du backbone ne suit pas une loi exponentielle.

L'une des études les plus importantes montre que le trafic Internet, aussi bien en Ethernet qu'en réseaux étendus, a les propriétés d'un trafic auto-similaire (ou fractal). Le phénomène d'auto-similarité stipule que nous pouvons décrire statistiquement le trafic Internet par des structures similaires, d'où viennent les notions de dépendance à long terme (*Long-Range Dependence* en anglais) et de corrélation.

En effet, le processus de Poisson considère que le processus d'arrivée est aléatoire et que les arrivées sont indépendantes alors qu'en réalité, l'arrivée des paquets dans une même session présente une forte corrélation entre les temps d'inter-arrivée des paquets. Dans ce même contexte, l'étude de (Paxson and Floyd, 1995; Fischer and Masi, 2006) met en évidence le facteur d'auto-corrélation entre les paquets, ce qui amène à rejeter l'hypothèse de l'indépendance entre les arrivées. Cette réalité a donné naissance à de nouvelles études pour trouver des distributions et des lois mathématiques qui modélisent mieux le processus d'arrivée au niveau paquet.

L'analyse pionnière (Paxson and Floyd, 1995) a attiré l'attention sur ce problème, en faisant l'analyse du processus d'arrivée des paquets d'un trafic TCP au niveau couche réseau. Les résultats de la distribution des inter-arrivées ont été comparés à la distribution exponentielle à l'échelle logarithmique. Cette comparaison montre qu'il n'y a correspondance entre les courbes qu'au-dessus de 0.1s, mais, ailleurs la distribution exponentielle sous-estime la distribution réelle. Des études similaires pour le trafic Web (Crovella and Bestavros, 1997)

montrent que la distribution du temps d'inter-arrivée des paquets suit une loi Pareto, de la famille des distribution à queue lourde.

Dans la même perspective, l'étude de la fonction de densité de probabilité du temps d'inter-arrivée des paquets dans des réseaux locaux sans fil ou dans des réseaux dédiés aux centres de données (Liang et al., 2011; Benson et al., 2010) montre qu'elle est modélisée par les lois Pareto, Weibull ou Log-normale, selon le réseau et les applications.

De plus, une analyse d'un trafic BitTorrent dans les réseaux IPv4 et IPv6, le protocole le plus populaire des échanges Point-à-Point, est faite par (Cifliklia et al., 2010) et montre l'aspect de l'auto-similarité du trafic en terme de temps d'inter-arrivée et la taille des paquets. La même étude montre en plus que les fonctions de densité de probabilité du temps d'inter-arrivée qui s'accordent plus aux traces réelles sont la distribution Weibull pour IPv4 et la distribution Gamma pour IPv6.

Dans cette section nous avons présenté une revue de littérature de la majorité des aspects liés au sujet de calcul de la gigue. Ceci nous permet de déduire qu'il est important de généraliser le modèle de calcul de la gigue proposé par (Dahmouni et al., 2012a) pour couvrir des files d'attente avec des processus d'arrivée et de service plus réalistes et plus précis, vu l'échec du modèle M/M/1. D'ailleurs, cela nous permettra de déterminer la méthodologie du travail et l'environnement à utiliser. Le chapitre (4) leur est consacré.

## 2.2 Routage et ingénierie de trafic

De nombreux travaux ont porté sur la planification des réseaux ATM et MPLS (Sansò and Soriano, 1998; Awduche, 1999) car l'ingénierie du trafic représente un sujet très important pour le transport des flux en garantissant la bonne qualité des applications. Il s'agit de calculer l'écoulement du trafic sur les liens du réseau afin de réaliser une performance spécifique. D'autres travaux, comme ceux de (Chen and Nahrstedt, 1998; Apostolopoulos et al., 1998), introduisent le routage orienté QoS, qui a pour objectifs de trouver les routes qui satisfont les contraintes de la qualité de service et de maximiser l'efficacité de l'utilisation des ressources du réseau.

On trouve aussi dans (Hoang, 2007) une formulation opérationnelle du problème de l'optimisation du routage et l'affectation des capacités qui se fonde sur la minimisation du délai de transmission de bout-en-bout et l'ajustement de la bande passante nécessaire pour une application.

Récemment, une méthode d'optimisation d'un trafic multiplexé dans des tunnels MPLS est donnée par (Srivastava et al., 2009) dans le but de minimiser la distorsion du trafic et de

garder une bonne qualité de service pour tous les flux hétérogènes qui composent le trafic. Elle offre une formulation en termes de programmation quadratique en nombres entiers, dans laquelle la fonction objectif prend en considération le délai et la gigue, chacun avec une pondération. Les contraintes se résument dans la restriction sur la capacité des liens et la limitation du nombre des tunnels actifs.

La nécessité de contrôler la gigue pour la majorité des services IP laisse les chercheurs actuels se concentrer sur la gestion du routage basé sur le délai et la gigue. L'une de ces recherches est suggérée par (Dahmouni et al., 2012b) et elle présente, en fait, une application de (Dahmouni et al., 2012a). Le problème de routage général, qui consiste à trouver l'ensemble des chemins de flux $X_p^{O,D}$ pour minimiser la combinaison linéaire du délai et de la gigue, est donné comme suit

$$\min_X F(X_u^{O,D}) = (1-\alpha)D + \alpha J \tag{2.21}$$

$$\sum_p X_p^{O,D} = \lambda^{O,D} \tag{2.22}$$

$$J^{O,D} \leq \overline{J}^{O,D} \tag{2.23}$$

$$D^{O,D} \leq \overline{D}^{O,D} \tag{2.24}$$

$$X_p^{O,D} \geq 0 \tag{2.25}$$

Les différents paramètres qui interviennent dans la définition du programme sont

$(O, D)$ la paire Origine/Destination

$D$ Le délai moyen pondéré pour toutes les paires (O,D)

$J$ La gigue moyenne pondérée pour toutes les paires (O,D)

$X_u^{O,D}$ le volume du trafic (O,D) acheminé sur le lien $u$

$\lambda^{O,D}$ Le taux d'arrivée du trafic d'origine O et destination D

$J^{O,D}$ Le délai moyen pondéré par (O,D)

$D^{O,D}$ La gigue moyenne pondérée par (O,D)

$\overline{J}^{O,D}$ La limite supérieure du délai moyen pour les paquets (O,D)

$\overline{D}^{O,D}$ La limite supérieure de la gigue pour les paquets (O,D)

Cette modélisation permet d'examiner l'impact de la gigue dans les contraintes sur le routage optimal. Le résultat qualitatif de cet algorithme de routage est différent de ceux qui se basent sur le délai seulement.

Ces travaux montrent la pertinence de la modélisation analytique des paramètres de la QoS en général et la gigue en particulier. Sauf que, ces algorithmes reposant sur le modèle M/M/1,

ils ne représentent pas bien la réalité puisqu'on sait que le modèle de Poisson est inadéquat pour modéliser les flux sur Internet. De plus, le programme donné par (2.21) possède un temps de convergence très élevé pour un nombre de flux supérieur à deux, ce qui ne permet pas son utilisation dans un réseau.

## 2.3 Contrôle pour le trafic vidéo sur IP

### 2.3.1 Contrôle du tampon de gigue

Le tampon de gigue est un composant fondamental des services multimédia, particulièrement en vidéo, pour la transmission streaming unidirectionnelle ou pour la vidéo interactive. Il permet au récepteur de resynchroniser le trafic reçu au taux d'émission de la vidéo pour reconstruire un flux régulier. Deux technologies de tampon de gigue sont proposées pour les lecteurs de médias, avec une taille du tampon fixe ou adaptable.

Dans cette section, nous résumons les recherches les plus importantes sur le fonctionnement et l'ajustement des tampons de gigue. L'étude de ces tampons a commencé dès les années 80, pour une première analyse proposée par (Barberis and Pazzaglia, 1980) en introduisant un dispositif au récepteur qui ajoute un délai fixe pour les paquets de voix sur IP (VoIP) pour résoudre le problème de synchronisation. Ce délai déterministe ne s'additionne qu'au premier paquet reçu. C'est un délai additionnel appelé le tamponnage initial ou aussi retard au démarrage (*start-up delay* en anglais). Pour l'approche fixe, il y a encore l'étude (Verhelst and Roelands, 1993) qui détermine un délai de tamponnage initial en trouvant un compromis entre les pertes et le délai d'attente.

Selon le comportement variable du réseau, surtout pour les réseaux mobiles, cette approche n'est plus efficace. Plusieurs algorithmes dynamiques sont proposés pour l'ajustement de la taille en délai de tampons de gigue. Dans (Ramjee et al., 1994), la technique proposée est indépendante des pertes et se base sur une méthode de synchronisation absolue. La détermination du temps de lecture d'un paquet $i$ ($p_i$), dépend du paquet, selon qu'il est le premier d'une d'une rafale ou non.

Si le paquet $i$ est le premier arrivé d'une rafale, le temps qu'il va passer dans la mémoire du tampon de gigue est calculé par

$$p_i = t_i + \hat{d}_i + \beta \times \hat{v}_i, \tag{2.26}$$

où $\hat{d}_i$ et $\hat{v}_i$ sont des estimations de la moyenne et la variance du délai de transmission de bout-en-bout $d_i$ approprié au paquet $i$, qui est égale à la somme du délai de transmission sur

le réseau $T_i$ et le délai passé dans le tampon de gigue $p_i$ ($d_i = T_i + p_i$), et la valeur de $\beta$ par défaut est mise à 4.

Si le paquet $j$ appartient à cette même rafale,

$$p_j = p_i + t_e^j - t_e^i \tag{2.27}$$

où $t_e^i$ et $t_e^j$ sont les temps d'émission des paquets $i$ et $j$. Les estimations de $\hat{d}_i$ et $\hat{v}_i$ sont basées sur une méthode linéaire récursive

$$\hat{d}_i = \alpha \hat{d}_{i-1} + (1-\alpha) T_i, \tag{2.28}$$

$$\hat{v}_i = \alpha \hat{v}_{i-1} + (1-\alpha) \|\hat{d}_i - T_1\|, \tag{2.29}$$

où $\alpha$ est un facteur de pondération. D'autres auteurs se sont ensuite inspirés de cet algorithme. Certains proposent une valeur de $\alpha$ adaptative (Kansar and Karandikar, 2001). D'autres adoptent la technique de détection des pics du délai de transmission sur le réseau. La calcul de $\hat{d}_i$ devient

$$\hat{d}_i = \hat{d}_{i-1} + T_i - T_{i-1}. \tag{2.30}$$

Le terme $T_i - T_{i-1}$ n'est que la gigue en IPDV.

D'autres techniques qui tolère les pertes, comme l'algorithme *Concord* (Sreenan et al., 2000), utilisent un histogramme du délai de transmission $T_i$ observé sur un échantillon de paquets dans le but de construire la distribution approximative de $T_i$. L'algorithme dans ce cas définit pour chaque flux $s$ deux seuils : le maximum de délai de transmission acceptable $T_s^{max}$ et le maximum du retard par paquet $MRP_s$. Le problème revient à trouver dynamiquement la taille du tampon de gigue minimum $p_i$ qui satisfait les contraintes

— $\forall i,\ T_i + p_i = d_i$.
— $d_i < T_s^{max}$
— $d_i$ ne conduit pas à des retards supérieur à $MRP_s$.

Des techniques plus récentes sont apparues depuis 2004, basées principalement sur la QoE (Narbutt and Murphy, 2004). L'approche suivie est la prédiction des taux de perte en se référant à la distribution du délai de transmission. On l'utilise dans un modèle d'optimisation qui cherche le temps de lecture $p_i$ optimal qui maximise la qualité du multimédia en général.

Une autre méthode de contrôle des tampons de gigue utilise l'adaptation dynamique du taux de lecture $\mu(n)$ (Li et al., 2012), où $n$ est l'occupation du tampon. L'algorithme mesure la

gigue au niveau du réseau à partir du dernier paquet reçu, par la moyenne suivante

$$J_i = \frac{\sum_{y=1}^{i-1} |X_i - A|}{i-1} \tag{2.31}$$

$$= (1 - \frac{1}{i-1})J_{i-1} + \frac{|X_{i-1} - A|}{n-1}, \tag{2.32}$$

où $X_i$ est le temps d'inter-arrivée au tampon de gigue du paquet $i$ et $A$ est le délai du tamponnage normalisé. Ensuite, il calcule un seuil $P_n$ selon quelques paramètres de design $L_0$ et $c$ avec l'équation

$$P_n = L_0 + (L - L_0) \min\{1, \frac{J_i}{c \times A}\}. \tag{2.33}$$

De plus, il définit pour le tampon de gigue un seuil d'accélération $P$ et un seuil de ralentissement $L$. Le taux de lecture approprié devient

$$\mu(i,n) = \begin{cases} R_L(n) & \text{si } n < L \\ R_S(i) & \text{si } L \leq n \leq H \\ R_H(n) & \text{si } n > H, \end{cases} \tag{2.34}$$

où $R_S$ est équivalent au taux d'arrivée, $R_L$ et $R_H$ sont des taux de lecture calculés à partir de deux coefficients : l'un pour diminuer le taux de lecture actuelle et l'autre pour l'accélérer.

Cette section donne un aperçu global sur les différents aspects du contrôle des tampons de gigue dans la littérature. Cela nous aide à positionner notre contribution par rapport à ce qui existe déjà. En effet, les algorithmes proposés par (Ramjee et al., 1994; Kansar and Karandikar, 2001) sont plus adéquats à un trafic ON-OFF tel que le trafic de la voix sur IP, ce qui n'est pas le cas pour la vidéo. La technique utilisée par l'algorithme *Concord* n'est pas toujours applicable surtout en cas du trafic UDP, où il n'est pas possible d'estimer le délai de transmission des paquets. Les approches qui se basent sur la QoE estiment généralement les performances du tampon ou son occupation en utilisant une modélisation en Poisson. Il s'est avéré que ce modèle n'est pas réaliste pour modéliser le trafic vidéo, surtout qu'il n'y a pas beaucoup de multiplexage à la réception .

### 2.3.2 Transmission adaptative de la vidéo

La consommation élevée de la bande passante par la transmission vidéo exige des fournisseurs de service d'adopter des techniques d'ajustement de débit de la transmission afin de bien gérer les ressources et contrôler la congestion. L'objectif majeur est toujours l'amélioration de la QoE pour satisfaire les utilisateurs. Différentes techniques sont proposées dans

la littérature mais leur but principal est d'adapter le débit vidéo au débit du canal. Deux groupes d'algorithmes sont développés pour les applications RTP/UDP et pour les applications HTTP/TCP. Il s'agit d'algorithmes de la couche application.

**Contrôle de congestion sur RTP/UDP**

L'un des algorithmes les plus importants est celui de *Google Congestion Control* (GCC) développé pour la norme *WebRTC* (De Cicco et al., 2013). Il permet d'adapter dynamiquement le débit vidéo à la condition instantanée du réseau. Il repose sur une composition hybride qui fait intervenir l'émetteur et le récepteur. La partie de contrôle côté récepteur, commence par un filtre de *Kalman* et son rôle est de déterminer pour chaque paquet reçu la mesure

$$d(i) = (t_r(i) - t_r(i-1)) - (t_e(i) - t_e(i-1)). \tag{2.35}$$

Nous rappelons que $t_e$ et $t_r$ sont les instants où le paquet a quitté l'émetteur et est arrivé au récepteur. Cette mesure n'est que la gigue IPDV. L'estimation du délai de transmission d'une trame de taille $L$ sur un lien de capacité $C$ est $\hat{d}_i = L/C$ et l'équation (2.35) devient

$$d(i) = \frac{L(i) - L(i-1)}{C} + w(i) = \frac{dL(i)}{C} + w(i), \tag{2.36}$$

où $w(i)$ est la variable aléatoire d'un bruit blanc gaussien. Cette variable est définie par le GCC comme la somme de la variation du temps d'attente $m(i)$ dans les différents nœuds du réseau et la gigue réseau $n(i)$. Le filtre envoie une métrique seuil à l'unité de *Detecteur de la surutilisation*. Cette entité décide de l'état du réseau : la bande passante est en état normal, sur- ou sous- utilisée. Cette comparaison se base sur $m(i)$. L'étape suivante, le *Contrôle de débit distant*, consiste à calculer le débit suggéré à l'émetteur $A_r(i)$ quand le $i^{ième}$ groupe de paquets RTP est arrivé. Elle se fait selon le système d'équation

$$A_r(i) = \begin{cases} \eta A_r(i-1), \text{ augmentation, avec } \eta \in [[1.005, 1.3] \\ \alpha R(i), \text{ dimunution, avec } \alpha \in [0.8, 0.95] \\ A(i-1), \text{ pas de changement,} \end{cases} \tag{2.37}$$

où $R(i)$ est le débit de reception. Cette information de contrôle est envoyée par la suite dans un message de feed-back usuel de ce protocol *Real-time Transport Control Protocol* (RTCP). La partie de contrôle côté émetteur utilise un algorithme basé sur le taux de perte des paquets $f_l(k)$ et il agit à chaque fois que le $k^{ième}$ RTCP message est reçu. La bande passante du côté

émetteur $A_s$ est alors calculée par

$$A_s(k) = \begin{cases} \max(X(k), A_s(k-1) \text{ si } f_l(k) > 0.1 \\ 1.05 \times (A_s(k-1) + 1) \text{ si } f_l(k) < 0.02 \\ A_s(k-1) \text{ sinon.} \end{cases} \tag{2.38}$$

La décision finale pour le nouveau débit d'envoi adapté à la bande passante disponible est donnée par

$$\hat{A}_s(k) = \min(A_s(k), A_r(k)). \tag{2.39}$$

Malgré la fiabilité de cet algorithme, l'échange fréquent des rapports de contrôle peut dégrader la qualité de transmission, surtout quand le réseau est déjà en cas de congestion.

Un autre algorithme récent est présenté par (NagyM. et al., 2014), destiné aux communications vidéo interactives. L'idée générale du *FEC-based Rate Adaptation* (FBRA) est d'activer un flux *Forward error correction* (FEC) pour qu'il joue le rôle d'une sonde pour estimer la bande passante disponible. Il fonctionne en ajustant le débit d'envoi au débit du flux FEC selon les étapes suivantes

- État *Stay* (absence de congestion) : l'algorithme n'active pas le flux FEC et ne modifie pas le débit d'envoi.
- État *Probe* (test de congestion) : l'algorithme active le flux FEC, maintient un débit constant et attend un signe de congestion à partir des rapports RTCP.
- État *Up* : le flux FEC est changé par un flux vidéo afin d'améliorer la qualité de la vidéo. S'il y a une détection de congestion, l'algorithme passe à l'état *Down*, sinon il passe à l'état *Stay*.
- État *Down* : Le débit d'envoi de la vidéo est réduit quand le RTCP rapporte une congestion. L'algorithme passe à l'état *Stay* quand il n'y a plus de congestion, sinon il reste sur le même état.

L'algorithme fournit une bonne qualité de transmission et une bonne adaptation par rapport à la condition du réseau, sauf qu'il génère beaucoup de trafic, ce qui peut être problématique, surtout dans les réseaux d'accès.

### Adaptation de la qualité de la vidéo sur HTTP/TCP

Les algorithmes d'adaptation de la qualité de la vidéo sur HTTP ( *Dynamic Adaptive Streaming over HTTP (DASH) en anglais*) sont plus populaires et utilisés par plusieurs fournisseurs du streaming vidéo. Nous donnons dans cette sous-section un aperçu de cette approche.

Le protocole de streaming adaptatif est standardisé par *3GPP* sous l'appellation *3GP-DASH*. La spécification donne une présentation de la structure des données accessibles par DASH, la normalisation du format du segment qui présente l'unité des données et la description du protocole du transfert. La présentation des médias est une structure de données encodée sous un format spécifique et présente une ou plusieurs périodes disjointes. Chaque période contient une ou plusieurs représentations pour le même média. La représentation est une séquence d'un ou plusieurs segments. Le segment contient les *métadonnées*. L'implémentation typique de cet algorithme découpe chaque source vidéo/audio en plusieurs segments (*chunks* en anglais) et les encodes dans un format désirés. Pour le codéc vidéo, cela signifie que chaque segment est découpée à la frontière de chaque groupement d'image de la vidéo (*Groupe of Pictures (GoP)* en anglais). Ce découpage permet l'encodage de chaque segment indépendamment des autres segments. Le délai de chaque segment peut aller de 2s jusqu'à 10s. Les segments encodés sont hébergés chez un serveur HTTP. Le client demande un segment du serveur d'une manière linéaire et il le télécharge avec un streaming progressif sur HTTP. La partie adaptation intervient quand cette source vidéo/audio est encodée en plusieurs débits correspondant à différents niveaux de qualité $l_n$. Cette technique est appelée l'encodage évolutif de la vidéo (*Scalable Video Coding* (SVC) en anglais). Ce type d'encodage est déjà fourni par MPEG-4 et le H.264.

C'est au client par la suite de décider quel débit choisir ($l_n \in \{1, 2, 3..., L_m\}$) selon la bande passante disponible pour lui. Dans la littérature, plusieurs algorithmes sont proposés pour que le lecteur détermine le débit/qualité de la vidéo qui convient à sa bande passante. Il est possible de les regrouper en deux familles, selon l'estimation de la bande passante ou selon l'état du tampon de gigue. *Throughput-based group* utilise principalement une prédiction de la bande passante disponible pour l'utilisateur, et cette mesure $\hat{A}_i$ est mise à jour pour chaque segment $i$ téléchargé

$$\hat{A}_i = (1 - \delta)\hat{A}_{i-1} + \delta A_i, \tag{2.40}$$

où $A_i$ est la bande passante instantanée et $\delta$ un coefficient de déviation.

*Buffer-based group* introduit deux techniques principales pour permettre l'adaptation dynamique du niveau de qualité par le client : le contrôle de l'occupation du tampon de gigue ou de la QoE. La première catégorie utilise deux seuils pour l'occupation du tampon afin de définir trois zones : une zone de fonctionnement normal, généralement intermédiaire, une zone critique où le tampon va se vider prochainement et une zone stable où le tampon est loin d'être vidé. Selon le niveau de l'état du tampon, le serveur réagit en augmentant, diminuant ou maintenant le niveau de la qualité de la vidéo. La deuxième technique fait appel à l'approximation de la QoE en termes de probabilité de vidage du tampon (*underflow*). Le

calcul de cette probabilité $p_u$ se base sur le principe de grande déviation pour compter les évènements où la mémoire du tampon de gigue est vide. La modélisation de ce tampon est donnée par une file d'attente M/D/1, ce qui donne (Chen et al., 2016)

$$p_u \approx \exp(-NL(a_u)), \tag{2.41}$$

où $N$ est la période du temps des prochains arrivées, $a_u$ dénote la réduction moyenne de l'occupation du tampon et $L(.)$ est une fonction définie pour satisfaire le principe de grande déviation. L'algorithme fait appel au calcul de $p_u$ seulement lorsque l'occupation de la file d'attente du tampon de gigue est supérieure à un seuil initialement déterminé. De cette manière, l'algorithme tend à limiter la fréquence des changements de niveaux.

Toutes ces approches du DASH exécutent des algorithmes du côté client. La technique de l'adaptation fait appel à un échange de requête de demande/réponse. Ceci charge les réseaux avec des flux de contrôle à part ceux qui sont déjà générés par TCP.

Ces techniques proposées pour RTP ou TCP se basent essentiellement sur un traitement pour la sélection du débit convenable de la source. Ce traitement est généralement effectué du côté du client. Ceci génère deux problèmes majeurs. Le premier est lié à la consommation de l'énergie des terminaux, surtout en cas des équipements mobiles. Le deuxième est le trafic de contrôle additionnel pour rapporter la qualité de la vidéo sélectionnée au serveur. Une solution à ces problèmes est d'intégrer le serveur dans ce contrôle qui fait l'objet de la section 7.

# CHAPITRE 3 DÉMARCHE DE L'ENSEMBLE DU TRAVAIL DE RECHERCHE ET ORGANISATION GÉNÉRALE DU DOCUMENT INDIQUANT LA COHÉRENCE DES ARTICLES PAR RAPPORT AUX OBJECTIFS DE LA RECHERCHE

Cette thèse, soumise sous forme d'une thèse par article, comprend neuf chapitres dont quatre sont des articles. Dans le but de présenter le travail accompli ainsi que ses contributions, la suite de cette thèse est structurée comme suit :

Le chapitre 4 présente l'article *Calculation of Packet Jitter for Non-Poisson Traffic* publié dans le journal *Annals of Telecommunications*. Il comprend les formalisations proposées pour le calcul de la gigue dans le cas d'un trafic non-Poisson pour un nœud du réseau, en plus de la validation des modèles introduits avec les simulations et l'évaluation du temps de calcul pour chaque modèle proposé.

Le chapitre 5 correspond à l'article *On the Relationship between Packet Jitter and Buffer Loss Probabilities* présenté à la $17^{ième}$ édition de la conférence Networks de IEEE. Il étudie la corrélation entre la gigue calculée dans un tampon à gigue et la performance de ce tampon en termes de probabilité d'échec (Buffer Overflow Probability et Buffer Underflow Probability en anglais). Il aborde l'estimation analytique de cette probabilité et la possibilité d'utiliser les modèles d'estimation de la gigue en cas de file d'attente infinie pour les tampons de gigue.

Le chapitre 6 est l'article *End-to-End Network Queuing Model Equivalent for Video Applications* soumis dans le journal *Computer Networks*. Il introduit une nouvelle modélisation d'une connexion de transmission vidéo, autant pour le streaming sur TCP que la vidéo interactive sur UDP. Cette modélisation est caractérisée et validée avec l'estimation de la gigue, ce qui donne un algorithme qui fait analytiquement le calcul de la gigue du côté serveur.

Le chapitre 7 présente l'article *A Server-Side Adaptive Control for Video Streaming* soumis dans le journal *IEEE Transactions on Multimedia*. Il s'agit d'une application de l'estimation de la gigue réseau effectuée par l'émetteur et elle présente également une méthode pour optimiser l'utilisation des informations qui circulent sur le réseau. Dans ce contexte, ce chapitre décrit la méthodologie pour monter un système pour l'adaptation dynamique de la qualité de la vidéo du côté serveur, ainsi que la validation de son fonctionnement et sa comparaison avec un algorithme d'adaptation populaire du coté client tel que celui utilisé par *Microsoft Silverlight* adopté par *Netflix*.

Le chapitre 8 contient une discussion générale sur les contributions et les limites des travaux. Le chapitre 9 est dédié à une conclusion générale de la thèse.

Nous discutons aussi dans ce chapitre la cohérence entre les articles scientifiques et les objectifs de notre recherche. Cette thèse a pour objectif d'étudier le paramètre de la gigue en proposant des modélisations mathématiques, en plus d'appliquer ces modèles pour améliorer la qualité de la transmission vidéo sur les réseaux Internet. Pour atteindre ces objectifs, nous séparons l'étude en deux grandes parties : d'une part, nous étudions la gigue dans le cas d'un seul nœud et son application pour contrôler les tampons de gigue, d'autre part, nous traitons le cas de la gigue sur un réseau IP de bout-en-bout et son application pour le transfert adaptatif de la vidéo sur HTTP. Les deux points entre lesquels nous mesurons la gigue pour chacun des deux cas d'études sont différents. La figure 3.1 identifie l'emplacement de la mesure de la gigue pour ces deux cas, ainsi que les parties du réseau où ces mesures s'appliquent.

En premier lieu, la formalisation mathématique de la gigue en cas d'un seul nœud, décrite dans le chapitre 4, peut être utilisée en pratique au niveau du tampon de gigue. Ce dernier présente un bon exemple d'un modèle par une seule file d'attente avec un seul flux à l'entrée. L'idée est d'appliquer la mesure de la gigue effectuée entre l'arrivée et le départ du tampon de gigue (voir figure 3.1) afin de contrôler les performances reliées à la QoE. Ceci fait l'objet du chapitre 5. Ceci est d'une importance particulière, puisque le client devient capable par lui-même d'estimer analytiquement les probabilités de pertes et de vidage du tampon. De plus, cette application aborde la problématique de l'utilisation de la gigue pour le contrôle des tampons de gigue.

En deuxième lieu, nous formulons un modèle mathématique pour la gigue réseau, mesurée entre l'arrivée au réseau du côté de l'émetteur et l'arrivée au tampon de gigue du côté du récepteur (voir figure 3.1). Contrairement au premier cas, ici, c'est au serveur d'estimer la gigue au niveau du réseau. Ce travail est discuté dans le chapitre 6. En outre, cette mesure présente une bonne métrique de correspondance entre la condition du réseau et la QoE. Pour cette raison, nous présentons dans le chapitre 7 une application de la modélisation analytique de la gigue réseau. Elle servira à contrôler la transmission de la vidéo du côté du serveur.

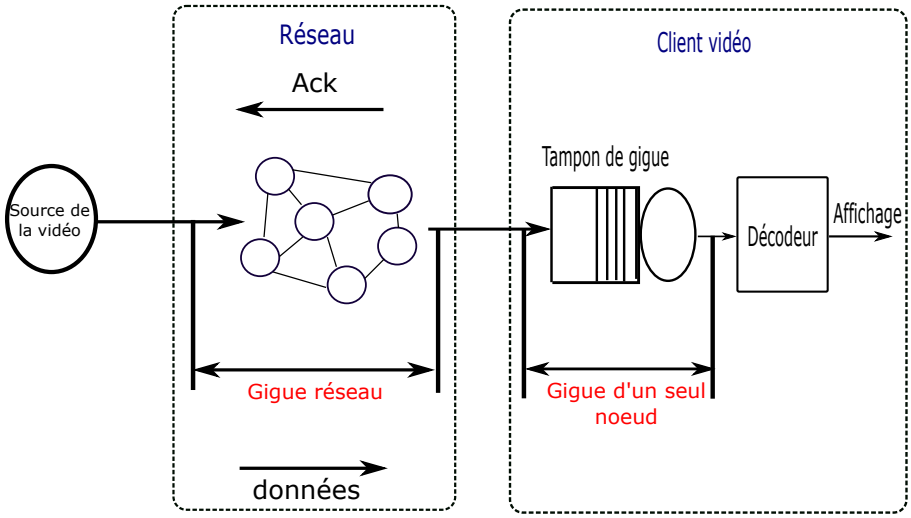



Figure 3.1 Emplacement des mesures de la gigue

# CHAPITRE 4 ARTICLE 1 : CALCULATION OF PACKET JITTER FOR NON-POISSON TRAFFIC

Recopié avec permission, H. Dbira, A. Girard et B. Sansò, *Calculation of packet jitter for non-poisson traffic*, accepté et publié aux Annals of Telecommunications, 2014 - Springer.

**Abstract**

The packet delay variation, commonly called delay jitter, is an important quality of service parameter in IP networks especially for real-time applications. In this paper, we propose exact and approximate models to compute the jitter for some non-Poisson FCFS queues with a single flow that are important for recent IP network. We show that the approximate models are sufficiently accurate for design purposes. We also show that these models can be computed sufficiently fast to be usable within some iterative procedure, e.g., for dimensioning a playback buffer or for flow assignment in a network.

## 4.1 Introduction

The Internet is now the main medium for a large number of applications and multimedia services. These applications require a good level of quality of service (QoS) simply to provide an adequate quality of experience to the customer. This is a challenge for network operators who must provide good transmission while dealing with fast-changing technology and increasing traffic growth so that computing QoS parameters quickly and accurately is a very important issue.

Usually, IP network planning and design use standard metrics based on queuing theory like the average delay and packet loss to optimize the network cost and performance (Hoang, 2007; Barreto and Carvalho, 2008). On the other hand, there is little work on computing the packet delay variation, also known as *jitter*, in IP networks. But this measure of QoS is particularly important for real-time services such as video conferencing, VoIP or video streaming and can have a greater impact on the user experience than latency and packet loss.

Standard bodies give different definitions of jitter. The IETF uses the mean absolute value of the packet delay variation (Demichelis and Chimento, 2002) while the ITU-T used the variation of delay from some minimum value in the stream (itu, 2011). Irrespective of the particular definition used, a basic difficulty is that delay jitter is based on the distribution

of packet delay, also called transit time, something that is often difficult to compute or even simply not known.

This is not a problem for M/M/1 queues where the transit time distribution is known. This has been used in a recent study (Dahmouni et al., 2012a) based on the IETF definition where a general definition of jitter is evaluated for an isolated queue and an approximate model is presented for a network of M/M/1 queues. Because the M/M/1 queue is not a good model for IP traffic, we want to extend these results to other queues.

In this paper, we focus on calculating the jitter, as defined by the IETF, in a single FCFS queue with a single flow but with non-Poisson processes. We make use of the transit time distribution when it is known to reduce the computation complexity. We also provide fast and accurate approximations when the transit time is unknown.

We also pay attention to the computation times since this is important for use in iterative procedures, say for optimizing the parameters of a queue or in a network design algorithm, or for controlling jitter for real-time applications.

In order to have some insight on jitter estimation in IP networks, we present in section 4.2 a brief literature review. The section also justifies our interest in jitter estimation for non Poisson queues. We recall in section 4.3 the general formula to compute jitter in a FCFS queue for a single flow of arbitrary statistics. This formula has two weak points. First, it depends on the distribution of the transit time, which is not known except for a limited number of queues. Also, it is in the form of a triple integral, which often requires long computation times and can suffer from poor convergence in some cases. We then propose in section 4.4 an exact analytic model for the G/M/1 queue where the transit time distribution is known. We show how it can be evaluated quickly by solving a simple nonlinear equation. We present in section 4.5 two important special cases for low and high load with arbitrary traffic, where the jitter does not depend at all on the transit time distribution so that we can derive exact formulas. Next, we use the high and low traffic values to propose approximations for two important classes of queues where the transit time is not known. We show in section 4.6 that a linear approximation is a very accurate model for the M/G/1 queue and we propose in section 4.7 a piece-wise linear approximation for the G/D/1 queue. In these two cases, the accuracy of the approximations is evaluated by comparing with simulation results. Finally, we present in section 4.8 a sample of the computation requirements for various processes.

## 4.2 Related Work

There has been much work in the last decades on the estimation of the delay jitter of ATM networks. The jitter distribution for a periodic traffic was derived in (Roberts and Guillemin, 1992) and has been used to shape traffic by dimensioning a leaky bucket. A complete characterization of the jitter for a constant bit rate traffic is provided by (Matragi et al., 1994, 1997). A similar analysis (Privalov and Sohraby, 1998) with periodic background traffic focuses on per-stream jitter. There is also some work on approximations for jitter in DiffServ networks. This is done for slot-based TDM traffic in (Chung and Soo, 2003) and for periodic arrivals with constant packet length in (Brun et al., 2006). Authors in (Alshaer and Elmirghani, 2008) computed the jitter generating function for a DiffServ queue where the EF traffic is an ON/OFF stream and the BE stream is Poisson.

This work cannot be used directly since ATM networks are quite different from IP networks. The packets are small and of constant length, the sources are generally modelled as periodic and the only measure of QoS is the cell loss probability, which has to be very low, typically less than $10^{-6}$ so that results can be derived based on the Chernoff bound. This is very different from IP, where packets have different lengths, the arrival processes are far from periodic and the QoS requirements are based on delay, loss, jitter and bandwidth.

Recently, a simple formula has been proposed (Dahmouni et al., 2012a) for computing the jitter both for a single queue and multiple queues in tandem for Poisson packet arrivals and exponential holding times. Several approximations are presented for a single queue for small, large and intermediate arrival rates of a tagged stream.

These models are then used in (Dahmouni et al., 2012b) to evaluate the effect of jitter on network routing. They find the optimal routing of IP packets subject to jitter constraints and evaluate the impact on network performance of taking jitter into account. The authors claimed that even though the Poisson model might not be very accurate in general, it *is* realistic in some kinds of access networks. Also, the accuracy of the *difference* between the two cases, with and without jitter constraints, might be good enough to draw some conclusions.

Still, there is a large body of work (Paxson and Floyd, 1995; Garcia et al., 2007) showing that packet traffic in local and wide area Internet networks, and especially traffic from real-time applications, is definitely not Poisson. The study of the Internet traffic in (Paxson and Floyd, 1995; Crovella and Bestavros, 1997) has shown that it can be modelled by a self-similar process and found that the inter-arrival distribution is best represented by heavy-tailed distributions. For WLANs, the study of packet inter-arrival time (Liang et al., 2011) has shown it to be more consistent with a Pareto, Weibull or Lognormal distribution. An

analysis for BitTorrent traffic through IPv4 and IPv6 networks is given by (Cifliklia et al., 2010) and shows that the distribution of inter-arrival time is Weibull in IPv4 and Gamma in IPv6. There are also many cases where the packet length distribution is not exponential. The authors of (Fraleigh et al., 2003) show that packets in IP backbones have a trimodal distribution with one packet size corresponding to TCP acknowledgment and two packet sizes of 572 and 1500 bytes corresponding to the maximum transmission units. For all these reasons, we want to extend the previous results to non Poisson traffic processes.

In this perspective, recent work (Angrisani et al., 2013) proposes a cross-layer experimental jitter measurement method for a real streaming flow. Delay jitter is evaluated directly through a linear average or a root average of delay variation samples, which are gathered from the protocol analyzer. A main result of this work is to show a strong correlation between the jitter as defined by IETF and video quality degradation. This is another reason why we need a better understanding of the jitter and some fast computation techniques in realistic cases, which is precisely the object of this paper.

## 4.3 General Formula

First we define the notation we will be using later and then provide a general formula for computing the jitter.

### 4.3.1 Notation

Throughout the paper, we denote a random variable by an upper case symbol like $X$, its mean by $m = E[X]$, its standard deviation by $s$ and its variance by $v = s^2$. The standard deviation of a distribution is often presented by the variable $\sigma$. Given the many distributions in our paper, we have used $\sigma$ for the scale parameter and $s$ for the standard deviation, because these two quantities are different for most distribution, e.g., the truncated normal. We also use $s$ as the variable for the Laplace transform. The meaning should be clear from the context.

We write the corresponding pdf as $f_X(x; \mu, \sigma, \ldots)$ with the appropriate list of parameters. The cdf is written as $F_X(x; \mu, \sigma, \ldots)$ and the Laplace transform of $f_X$ as $\mathcal{F}_X(s; \mu, \sigma, \ldots)$. We use the simplified form $f_X(x)$ whenever the value of the parameters is clear from the context. We will be using a number of distributions with pdfs denoted as follows : More information on these distributions is contained in Appendix 4.11.

### 4.3.2 The Density Function of the Jitter

We give some definitions and the notation that will be used in the paper. We then expand on the results of (Dahmouni et al., 2012a) by giving a detailed derivation of the pdf of the jitter in addition to the triple integral formulation for the mean that was presented in the paper.

In this paper, we adopt the IETF (Demichelis and Chimento, 2002) definition of jitter. It is based on the transit delay of successive packets between two measurement points. For a single queue, these are the entry into the buffer and the exit from the server. First define for packet $i$

$t_i$ arrival time,

$R_i$ Inter-arrival time, $R_i = t_i - t_{i-1}$

$\lambda$ Average arrival rate $\lambda = 1/E[R_i]$

$r_i$ departure time,

$W_i$ waiting time,

$S_i$ service time,

$\mu$ average service rate $\mu = 1/E[S_i]$

$T_i$ transit time, $T_i = W_i + S_i$,

$\eta$ average transit rate $= 1/E[T_i]$

We define the jitter as the random variable $J_i$ which is the absolute value of the difference in transit times of packet $i+1$ and $i$

$$J_{i+1} = |T_{i+1} - T_i| . \tag{4.1}$$

We then have

$$J_{i+1} = |T_{i+1} - T_i| = |W_{i+1} + S_{i+1} - (W_i + S_i)| \tag{4.2}$$

We make use of Lindley's recurrence, using the fact the condition that packet $i+1$ does not wait in the queue is expressed by $R_{i+1} \geq T_i$

$$W_{i+1} = \begin{cases} 0 & \text{if } R_{i+1} \geq T_i \ , \\ W_i + S_i - R_{i+1} & \text{otherwise} \end{cases} \tag{4.3}$$

so that we have

$$J_{i+1} = \begin{cases} |S_{i+1} - T_i| & \text{if } R_{i+1} \geq T_i \ , \\ |S_{i+1} - R_{i+1}| & \text{otherwise.} \end{cases} \tag{4.4}$$

Note that the random variables $T_i$, $S_{i+1}$ and $R_{i+1}$ are independent and also that their distribution is the same for all values of $i$. Define the probability density functions

$f_R(y)$ for the inter-arrival times

$f_S(z)$ for the service times

$f_T(x)$ for the transit times.

Next, we derive the pdf of $J_{i+1}$ expressed in terms of $f_R$, $f_S$ and $f_T$. We can write

$$\begin{aligned} & P\{J_{i+1} = w\} \\ = & \int P\{R_{i+1} = y, S_{i+1} = z, T_i = x\}\, dx\, dz\, dy \\ = & \int P\{R_{i+1} = y\}\, P\{S_{i+1} = z\}\, P\{T_i = x\}\, dx\, dz\, dy \end{aligned} \tag{4.5}$$

where we have used the fact that $R_{i+1}$, $S_{i+1}$ and $T_i$ are independent. The integral must be taked over all possible values of $y$, $z$ and $x$ that are compatible with the condition

$$w = \begin{cases} |z - x| & \text{if } y \geq x \\ |z - y| & \text{otherwise} \end{cases} \tag{4.6}$$

that simply reflect condition (4.4) on the radom variables themselves. We then have

$$\begin{aligned} & f_{J_{i+1}}(w) = \int_0^\infty f_{R_{i+1}}(y) \int_0^\infty f_{S_{i+1}}(z) \\ & [f_T(z - w)U(y - x) + f_T(z - y)U(x - y)]\, dz\, dy \end{aligned} \tag{4.7}$$

where we have defined the Heaviside function

$$U(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4.8}$$

Next, we compute the expectation

$$\begin{aligned} E[J_{i+1}] &= \int_{-\infty}^\infty w f_{J_{i+1}}(w) dw \\ &= \int_0^\infty f_{R_{i+1}}(y) \int_0^\infty f_{S_{i+1}}(z) h(y, z) dz\, dy \end{aligned} \tag{4.9}$$

where we have defined

$$h(y, z) = \int [w f_{T_i}(z - w)U(y - x) +$$

$$w f_{T_i}(z-y)U(x-y)]\,dx. \tag{4.10}$$

We can rewrite the inner integral over the domain of $x$ by noting from (4.6) that if $y \geq x$, then $w = |z-x|$ and that $w = |z-y|$ when $y < x$. Replacing in (4.9), we get

$$J = \int_0^\infty f_R(y) \left\{ \int_0^\infty f_S(z) \right. \\ \left. \left[ \int_0^y |z-x| f_T(x)dx + |z-y| \int_y^\infty f_T(x)dx \right] dz \right\} dy \tag{4.11}$$

where we have used the fact that all packets have the same $R$, $S$ and $T$ distributions to drop the index.

### 4.3.3 Our Contribution

It would seem that the problem of computing jitter is neatly solved by having the explicit expression (4.11). Unfortunately, things are not so easy for the following reasons. A first fundamental problem is that (4.11) requires the knowledge of $f_T$, the distribution of the transit time through the queue. In practice, transit time distributions are rarely known so that it is simply not possible to use the formula directly for most cases.

In the cases where $f_T$ is known, as for the G/M/1 queue, or can be approximated, it is quite unlikely that we can get a closed form solution for (4.11). In most cases, we would have to evaluate it numerically, which in turn raises the problem that it is always time consuming to compute a triple integral numerically. We want something simpler whenever speed is important.

The first contribution of the paper is thus to reduce the triple integral to a simple integral in some cases where the transit time distribution is known, which is much faster and yields accurate results.

In cases where the transit time is not known, the second contribution is to provide fast and accurate approximations for M/G/1 and G/D/1 queues, which are important in practice.

Finally, we also provide some very general exact results for arbitrary G/G/1 queues in both low and high traffic limits.

## 4.4 The G/M/1 Queue : Exact Value

The simplest case where we can get an exact value for the jitter is that of the G/M/1 queue with an exponential service time with parameter $\mu$. In this section, we emphasize the central

role played by the Laplace transform of the arrival distribution in the evaluation of the jitter. Computational issues are discussed in section 4.8.1.

We can get exact values for the jitter because we know (Kleinrock, 1975) that the transit time $T$ of a G/M/1 queue has an exponential distribution

$$f_T(x) = \eta e^{-\eta x} \tag{4.12}$$

$$\eta = \mu(1-\tau) \tag{4.13}$$

where $\eta$ is the average transit rate. $\tau$ is the probability of waiting and is given given by the unique root in the interval $(0,1)$ of

$$\tau = \mathcal{F}_R(\mu - \mu\tau) \tag{4.14}$$

where

$$\mathcal{F}_R(s) = \int_0^\infty f_R(y)e^{-sy}dy \tag{4.15}$$

is the Laplace transform of the inter-arrival distribution $R(y)$.

### 4.4.1 Simplified Model

Next, we use the fact that both the service and transit times are exponential to simplify (4.11) since the last two integrals can be computed explicitly. We get an expression for the jitter that depends only on the form of $R$ and the value of $\eta$.

$$\begin{aligned}
J = & \int_0^\infty f_R(y) \left\{ \int_0^\infty (\mu e^{-\mu s})ds \right. \\
& \left. \left[ \int_0^y |s-x|(\eta e^{-\eta x})dx + |s-y| \int_y^\infty (\eta e^{-\eta x})dx \right] \right\} dy \\
= & \int_0^\infty f_R(y) \left\{ \frac{2f_T(y)f_S(y)}{\eta\mu(\eta+\mu)} - \frac{f_T(y)}{\eta^2} + \right. \\
& \left. \frac{(\eta+\mu)}{\eta\mu} - \frac{2}{(\eta+\mu)} \right\} dy \\
= & \frac{(\eta^2+\mu^2)}{\eta\mu(\eta+\mu)} + \\
& \int_0^\infty f_R(y) \left\{ \frac{2f_T(y)f_S(y)}{\eta\mu(\eta+\mu)} - \frac{f_T(y)}{\eta^2} \right\} dy \\
= & \frac{(\eta^2+\mu^2)}{\eta\mu(\eta+\mu)} + \frac{2}{(\eta+\mu)} \int_0^\infty f_R(y)e^{-(\eta+\mu)y}dy - \\
& \frac{1}{\eta} \int_0^\infty f_R(y)e^{-\eta y}dy
\end{aligned}$$

$$= \frac{(\eta^2 + \mu^2)}{\eta\mu(\eta + \mu)} + \frac{2}{(\eta + \mu)}\mathcal{F}_R(\eta + \mu) - \frac{1}{\eta}\mathcal{F}_R(\eta) \tag{4.16}$$

where we have used (4.15) to replace the last two integrals and $\eta$ is the average transit rate given by (4.13).

### 4.4.2 Jitter vs Load for Some Distributions

In this subsection, we plot the jitter as a function of load for some important inter arrival time distributions. All the results are presented in units of service time. The load $\rho = \lambda/\mu$ is given on the horizontal axis and the jitter, measured in multiples of the service time, is plotted on the vertical axis. We also show that we can get an analytic form for $\mathcal{F}_R(s)$ for many $f_R$ distributions, which will simplify the numerical calculations.

#### Exponential Inter-arrival

We can get an exact formula (Dahmouni et al., 2012a) when the inter-arrival time is exponential

$$f_R(y) = \lambda e^{-\lambda y}. \tag{4.17}$$

In this case, the probability $\tau = \lambda/\mu$ so that the average transit rate becomes

$$\eta = \mu - \lambda. \tag{4.18}$$

We also get a closed form for the Laplace transform

$$\mathcal{F}_R(s) = \frac{\lambda}{\lambda + s}. \tag{4.19}$$

In that case, we can easily solve (4.14) and from this, we can compute the exact value from Eq. (4.16).

$$\begin{aligned} J &= E[\mid T_{j+1} - T_j \mid] \\ &= \frac{1}{\mu}. \end{aligned} \tag{4.20}$$

For the M/M/1 queue, the jitter is the average service time over the whole range of traffic.

### Deterministic Inter-arrival

We now consider a deterministic arrival process. This is useful to model some codecs that emit packets or frames at fixed intervals. In this case, the inter-arrival time is equal to a constant $m = 1/\lambda$ and its probability density function is

$$f_R(y) = \delta\,(y - m) \tag{4.21}$$

which gives

$$\mathcal{F}_R(s) = e^{-s/\lambda}. \tag{4.22}$$

In this case, even though we have a closed form for $\mathcal{F}_R(s)$, we cannot compute a closed form solution for (4.14). Still, replacing in (4.16), we get the jitter as a function of $\eta$ as

$$J_D = \frac{(\eta^2 + \mu^2)}{\eta\mu(\eta + \mu)} + \frac{2}{(\eta + \mu)} e^{-(\eta+\mu)/\lambda} - \frac{1}{\eta} e^{-\eta/\lambda} \tag{4.23}$$

The plot of the jitter as a function of the load is shown in Fig. 4.1. Note however that even with such a simple transform, it is not possible to solve (4.14) in closed form. This will be the case for all the distributions that we will be looking at.

### Gamma Inter-arrival

We can get a closed form of the Laplace transform when the arrival process follows a gamma distribution $Gm(k, \theta)$. The probability density function of the Gamma distribution $Gm(x; k, \theta)$ with shape $k > 0$ and scale $\theta > 0$ is given by (More details are in Appen-

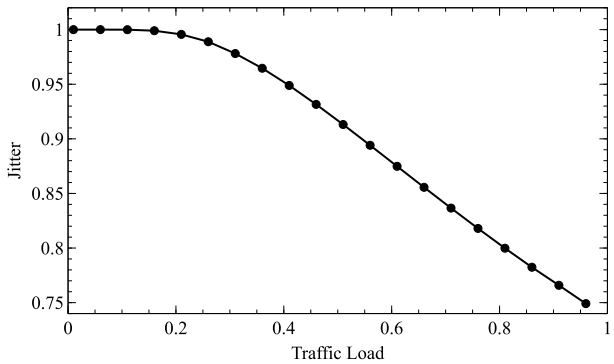



Figure 4.1 Jitter for a D/M/1 queue

dix. 4.11.1)

$$f_R(x) = x^{k-1}\frac{e^{-x/\theta}}{\theta^k \Gamma(k)}. \tag{4.24}$$

Replacing (4.24) in (4.15), we get

$$\mathcal{F}_R(s) = \frac{1}{(1+s\theta)^k} \tag{4.25}$$

from which we can get the value of the jitter from (4.16). We present in figure 4.2 the values of jitter computed for two values of the standard deviation $s = 0.5$ and 3. This shows that the variance of the process can have a significant effect of the jitter, as one would expect.

### Pareto Type I Inter-arrival

There is a large body of work (Leland et al., 1994; Paxson and Floyd, 1995) showing that the packet inter-arrival time in wide area or local networks is modeled by heavy-tailed distributions. One good model is the Pareto type I $P(x_m, \alpha)$ with scale $x_m$ and shape $\alpha$. It has a density function given by (More details in Appendix 4.11.2)

$$f_R(x) = \begin{cases} \alpha\dfrac{x_m^\alpha}{x^{\alpha+1}} & \text{if } x \geq x_m \\ 0 & \text{otherwise.} \end{cases} \tag{4.26}$$

We can compute (4.15) and get

$$\mathcal{F}_R(s) = \alpha E_{\alpha+1}(sx_m) \tag{4.27}$$

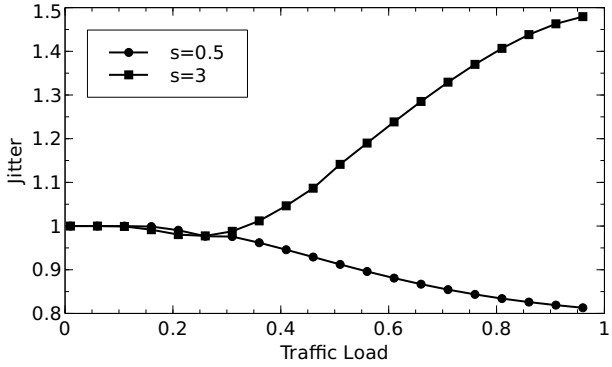



Figure 4.2 Jitter for a Gm/M/1 Queue

which we can replace in (4.16) to get the jitter. Here, $E_n(x)$ is the exponential integral

$$E_n(x) = \int_1^\infty \frac{e^{-xt}}{t^n} dt. \tag{4.28}$$

We present in figure 4.3 the value of the jitter as a function of load for the standard deviation $s = 0.25$.

## 4.5 The G/G/1 Queue : Exact Limit Values

There is another case where we can get exact results : For high and low traffic limits, the jitter does not depend on $T$. These results are interesting in their own right since they are valid for G/G/1 queues with a single flow so that we can use them to get insight on some important proprieties of jitter. Because they are relatively simple, it is possible to get either analytic formulas or some fast numerical evaluation. Also, as we discuss in sections 4.6 and 4.7, these limits will be used to build some approximations when $f_T$ is not known.

### 4.5.1 Small Arrival Rate

Suppose that the packet average arrival rate $\lambda$ is so low that packets arriving to the queue almost always find an empty queue. We then have $W_i \approx 0$ in (4.2) and in that case,

$$\lim_{\lambda \to 0} J = E\left[|S_{i+1} - S_i|\right]. \tag{4.29}$$

We then get the general result

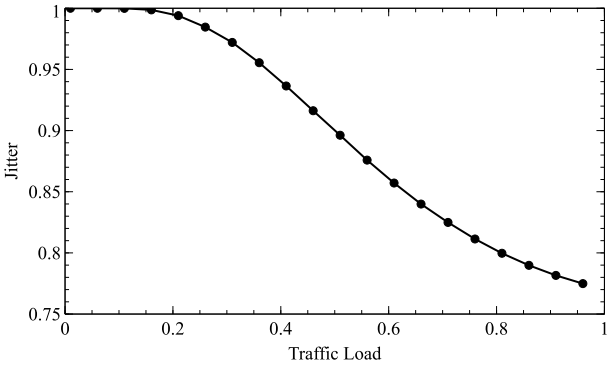



Figure 4.3 Jitter for a Pareto inter-arrival process

**Proposition 1** *For a G/G/1 queue, in the low-traffic limit, the jitter depends only on the service time distribution and is given by the expectation of the absolute value of the difference, or mean difference, of the service times of two consecutive packets.*

Note that for some queues, like batch arrivals, the condition $\lambda \to 0$ would apply to the arrival of batches. We cannot conclude that $W_i \approx 0$ and the result does not apply in these cases. Also, we see that at low load, jitter need not go to zero, which is very different from the behavior of delay which does go to zero at low load.

### 4.5.2 Large Arrival Rate

We can also get an approximation for high load when $\lambda \to \mu$. In this case, we assume that when packet $i$ arrives to the queue, packet $i-1$ has not yet started service. Let $\tau_i$ be the instant just before packet $i$ arrives to the queue and $W_i(\tau_i)$ the amount of time packet $i$ will have to wait, also called the workload, at time $\tau_i$. We have

$$T_i = W_i(\tau_i) + S_i. \tag{4.30}$$

Next, consider the queue just before packet $i-1$ arrives. The workload at that time is $W_{i-1}(\tau_{i-1})$. During the interval $R_i = t_i - t_{i-1}$, the workload will increase by $S_{i-1}$ since packet $i-1$ has joined the queue, and will decrease by $R_i$ since the server is busy during the whole interval from $\tau_{i-1}$ to $\tau_i$. The workload just before packet $i$ arrives is then given by

$$W_i(\tau_i) = W_{i-1}(\tau_{i-1}) - R_i + S_{i-1}. \tag{4.31}$$

Replacing (4.31) in (4.30), we get

$$\begin{aligned} T_i &= W_{i-1}(\tau_{i-1}) - R_i + S_{i-1} + S_i \\ &= T_{i-1} - R_i + S_i \end{aligned} \tag{4.32}$$

from which we get

$$\lim_{\lambda \to \mu} J = E\left[|S_i - R_i|\right]. \tag{4.33}$$

We then have the proposition

**Proposition 2** *For a G/G/1 queue, in the high traffic limit, the jitter is given by the expectation of the absolute value of the difference between the service and inter-arrival times of a packet.*

This is another important insight that shows how jitter remains finite at high load. This behavior is also very different from that of delay which becomes infinite at $\rho \approx 1$.

### 4.6 The M/G/1 Queue : Linear Approximation

It is known (Fraleigh et al., 2003) that the distribution of packet sizes in the Internet is generally not exponential so that we now present some models for these cases. We focus on M/G/1 queues where the service time follows a general distribution. It is well known that network traffic in the core does not generally have Poisson arrivals. Still, we have done measurements in the access portion of the network of a large ISP showing that the traffic arrival process behaves as Poisson after only a few levels of aggregation of connections. This is why we feel that it is important to examine the M/G/1 queue.

We know (Kleinrock, 1975) that the Laplace transform of the sojourn time distribution in a M/G/1 queue is given by

$$\mathcal{F}_T(s) = \mathcal{F}_S(s)\frac{s(1-\rho)}{(s-\lambda)+\lambda\mathcal{F}_S(s)}. \tag{4.34}$$

It would seem that we could use the same kind of simplification as for the G/M/1 queue of section (4.4). The general expression (4.11) could be simplified because the two inner integrals could be evaluated analytically, leaving only the integral over $y$. If we wanted to do this for the M/G/1 queue, we could derive (4.11) with $f_S$ as the outermost integral, then $f_T$ and $f_R$ as the innermost part. Because $f_R$ is an exponential, this inner integral could be evaluated analytically, but we would still have to evaluate a double integral. This is far more complicated than for the G/M/1 case for the following reasons.

In the simplest case, we can assume that we have an analytic expression for $\mathcal{F}_S$. We can then get an analytic expression for $\mathcal{F}_T(s)$ from (4.34). It is quite unlikely that we can compute a closed form solution for the inverse transform $f_T$. This means that every time we need a value for $f_T(x)$ in (4.11), we would have to compute numerically the inverse transform of $\mathcal{F}_T$, something that may not be all that easy to do since the numerical calculation of the inverse Laplace transform is notoriously difficult.

The overall computation will most likely be very time-consuming and instead, we propose a method based on linear interpolation using the fact that it is relatively easy to compute the asymptotic values (4.29) and (4.33). This is equivalent to the following proposition

**Assumption 1** *In an M/G/1 queue, the jitter is approximately linear as a function of the traffic load $\rho$.*

Under this assumption, all we need is the values for $J$ at $\rho = 0$ and $\rho = 1$ for which we can use the results of section 4.5. We present results for the M/D/1, M/Gm/1 and M/Nt/1 queues in figures 4.4, 4.5 and 4.6 where we check the accuracy of the linear assumption by comparing the calculated values, labeled `Line`, with the simulated values, labeled `sim`. The 95% confidence interval for the simulated values is smaller than the plot marker and does not show up on the graph. In all three cases, as well as for other distributions not shown here, we find that the agreement is excellent over the whole range of traffic.

To summarize, we see that for an M/G/1 queue, the linear approximation is very accurate for many different forms of the service time distribution. More importantly, this model is based on simple formulas for the two end points from section 4.5 that can sometimes be calculated analytically or, when this is not possible, can be evaluated numerically. In both cases, the approximation will be very helpful for any iterative procedure that needs to compute the jitter as a function of the load.

## 4.7 The G/D/1 Queue : Piece-wise Linear Approximation

We now examine queues with a constant service time since, as we mentioned in section 4.2, a significant amount of traffic is made up of packets of constant length (Fraleigh et al., 2003) so that

$$f_S(s) = \delta(s - \overline{S}). \tag{4.35}$$

Our first attempts were based on extending the techniques of the previous sections. First, we confirmed that using an exponential sojourn time $f_T(x)$ in (4.11) is not very accurate. The linear model that was such a good fit for the M/G/1 queue also turned out to be very inaccurate.

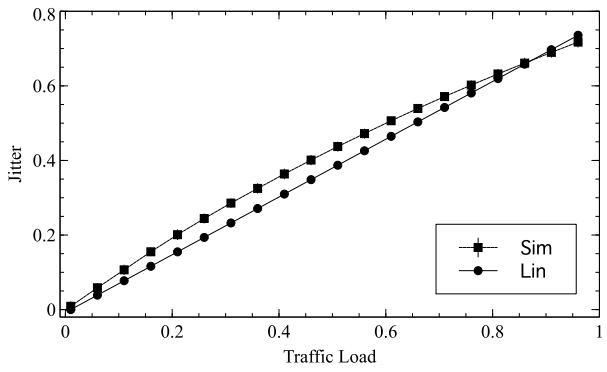



Figure 4.4 Jitter for a M/D/1 Queue

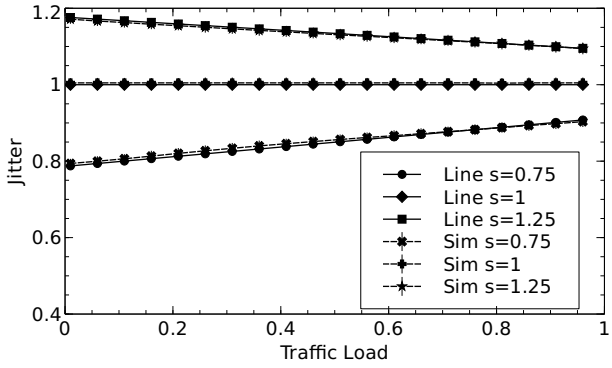



Figure 4.5 Jitter for a M/Gm/1 Queue

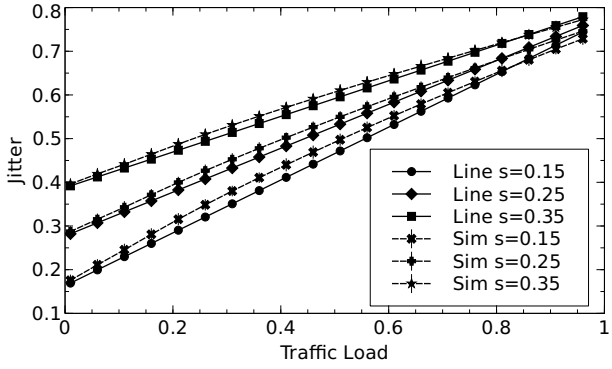



Figure 4.6 Jitter for M/Nt/1 Queue

We then ran a number of simulations whose results are presented in figures 4.7 and 4.8 and observed that the jitter is zero not only at very low load but for a significant range of low traffic values, sometimes as high as 0.5. After this point, we also saw that the jitter increases more or less linearly with load.

### 4.7.1 Two-Segment Approximation

This suggests an approximation based on two linear segments, one with zero slope and the other increasing linearly. We need to be able to compute three points $P_0$, $P_1$ and $P_2$, either analytically or numerically. We can use propositions (1) and (2) to compute the jitter for the two end points

$$P_0 = (0, J_0) \tag{4.36}$$

$$P_2 = (1, J_2) \tag{4.37}$$

where $J_0$ and $J_2$ are given by equations (4.29) and (4.33). Note that for the G/D/1 queue, $J_0 = 0$.

The intermediate point $\rho_1$ can be estimated based on the following argument. In the low-traffic limit, packets don't wait so that their transit time is simply the service time. This will happen as long as the interval between arrivals is significantly larger than the service time $\overline{S}$. More precisely, we consider that $J \approx 0$ as long as

$$Pr\left[R_i < \overline{S}\right] < \epsilon \tag{4.38}$$

for some small $\epsilon$, typically $10^{-2}$ and where $R_i$ is the inter-arrival time of packet $i$. The point $\rho_1$ is defined by solving (4.38) as an equality. Let $F_R$ be the cdf of $R$. The condition (4.38) can be written as

$$F_R(\overline{S}; m, s, \ldots) = \epsilon \tag{4.39}$$

which we can solve for $m$ and we get $\rho_1 = 1/(m\mu)$. From this, we can make a two-segment interpolation

$$J(\rho) = \begin{cases} 0 & \text{if } \rho \in [0, \rho_1] \\ \dfrac{J_2}{1-\rho_1}\left[\rho - \rho_1\right] & \text{if } \rho \in [\rho_1, 1]. \end{cases} \tag{4.40}$$

### 4.7.2 Three-Segment Approximation

Although the interpolation (4.40) is continuous in $[0, 1]$, its derivative is not continuous at $\rho = \rho_1$. This will be a problem if we want to use the formula within any algorithm that requires continuous derivatives wrt $\lambda$.

We now modify the technique of section 4.7.1 to get a continuous and differentiable function. First we choose a small $\delta > 0$, typically 0.1, and define two points close to $\rho_1$

$$\rho^+ = \rho_1(1 + \delta) \tag{4.41}$$

$$\rho^- = \rho_1(1 - \delta) \tag{4.42}$$

with the corresponding jitter values

$$J(\rho^-) = J_0 \tag{4.43}$$

$$J(\rho^+) = \frac{J_2}{1 - \rho_1}\left[\rho^+ - \rho_1\right] \tag{4.44}$$

where $J(\rho^+)$ is the value of $J$ at $\rho = \rho^+$ on the linear segment from (4.40).

The three segments are defined as follows. In the interval $[0, \rho^-]$, we take $J = J_0 = 0$. In the interval $[\rho^+, 1]$, $J$ is given by the linear approximation in the second part of (4.40). In the segment $[\rho^-, \rho^+]$, $J$ is modelled by a third degree polynomial

$$P(\rho) = a\rho^3 + b\rho^2 + c\rho^1 + d. \tag{4.45}$$

We choose the coefficients so that both the function and its derivative are continuous on the whole interval

$$P(\rho^-) = 0 \tag{4.46}$$

$$P'(\rho^-) = 0 \tag{4.47}$$

$$P(\rho^+) = \frac{J_2}{1 - \rho_1}\left[\rho^+ - \rho_1\right] \tag{4.48}$$

$$P'(\rho^+) = \frac{J_2}{1 - \rho_1}. \tag{4.49}$$

Conditions (4.46) and (4.48) guarantee the continuity at $\rho^-$ and $\rho^+$ and the two others ensure the differentiability. To summarize, the following 3-segments model is

$$J(\rho) = \begin{cases} 0 & \text{if } \rho \in [0, \rho^-] \\ P(\rho) & \text{if } \rho \in [\rho^-, \rho^+] \\ \dfrac{J_2}{1-\rho_1}\left[\rho^+ - \rho_1\right] & \text{if } \rho \in [\rho^+, 1]\ . \end{cases} \tag{4.50}$$

We check the accuracy of the model by comparing the values from (4.50) with simulation results for some distributions $f_R$ of the arrival interval.

### Gamma Inter-arrival

We start with the case where the inter-arrival time has a gamma distribution $Gm(k, \theta)$. Assume that the variance $s^2$ is given. In that case, the two parameters $k$ and $\theta$ depend only on the mean $m$ through (4.86–4.87). We can then write (4.39) as

$$F_R\left[\overline{S}, k(m), \theta(m)\right] = \epsilon \tag{4.51}$$

which we can solve numerically for $m$. From this value, we get $\rho_1 = \overline{S}/m$. Fig. 4.7 shows the results of the approximation with $\delta = 0.1$. Curves produced by simulations for different values of the variance are labelled `sim` and they are compared to the 3-segments curves labelled `Fit`. As in other cases, the agreement is very good for most cases, except when $s = 3$, which has a very large variance.

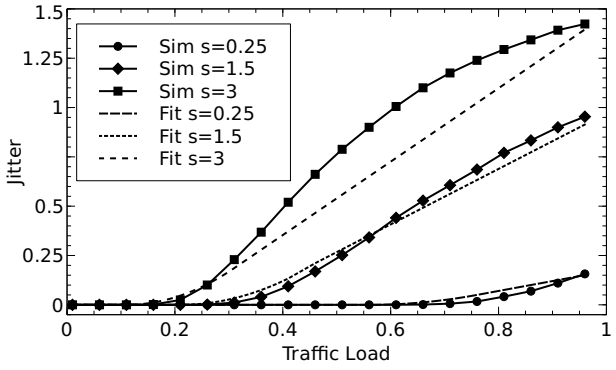



Figure 4.7 Jitter for Gm/D/1 Queue

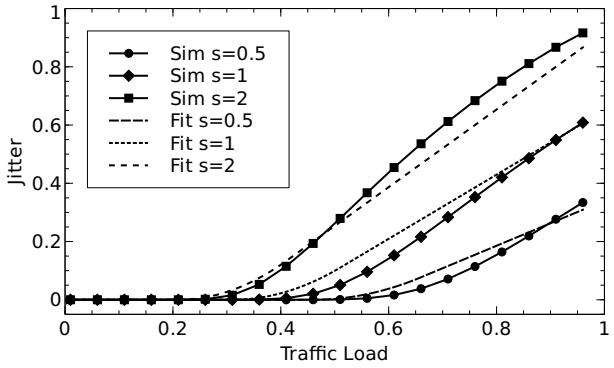



Figure 4.8 Jitter for LogN/D/1 Queue

### Log-normal Inter-arrival

We get similar results when the inter-arrival time follows a log-normal distribution. Its pdf $LogN(x, \mu, \sigma)$ is given by

$$f_X(x; \mu, \sigma) = \frac{1}{x\sqrt{2\pi}\sigma} \exp\left(-\frac{(\ln x - \mu)^2}{2\sigma^2}\right) \tag{4.52}$$

We compare the approximation to some simulation results in Fig. 4.8 for the same value of $\delta$ as for the Gm/D/1. Here again the agreement is found to be quite good.

## 4.8 Computation Times

The results of the previous sections show that it is possible to compute the jitter for some interesting classes of queues. We now discuss the computation times required for different traffic processes. We used the Python `SciPy` module for scientific calculations with default parameter settings. The only Python code was for calling the evaluation of the functions by the `numpy` module or the `stats` module of `SciPy`. The actual integration used the `quad` or `dblquad` function for single or double integration from the `QuadPack` module of `SciPy`. We computed the solution of nonlinear equations with the `fsolve` function from the `Scipy optimize` module. All these are the Netlib functions and are run as compiled code so that the cpu times reported here should not be too different from a full implementation in a compiled language. All calculations are done on an Intel core i5 with 2 cores at 2.53 GHz.

### 4.8.1 G/M/1

This is the simplest case since we have seen in section 4.4 that we can compute the jitter exactly for the G/M/1 queue. These results depend on the Laplace transform of the inter-arrival distribution $\mathcal{F}_R(s)$ in two ways. First, we have to compute $\mathcal{F}_R(\eta)$ and $\mathcal{F}_R(\eta + \mu)$ in (4.16). In order to do this, we also need to compute the value of the transit rate $\eta$ from (4.14) which also depends on $\mathcal{F}_R(s)$. We give a short discussion of the computation time required for these two parts

#### Computing the Average Transit Rate

This is potentially the more time-consuming part. The average transit rate is given implicitly by the solution of the nonlinear equation (4.14). Note however that in general, it is not possible to solve (4.14) analytically except for the M/M/1 queue. For instance, for the Gamma distribution, we have

$$\mathcal{F}_R(s) = (1 + \theta s)^{-k} \tag{4.53}$$

but replacing in (4.14) yields the nonlinear equation

$$\tau = \left[1 + \theta\mu\,(1 - \tau)\right]^{-k} \tag{4.54}$$

which cannot be solved analytically.

We have seen in section 4.4.2 that we can get an analytic expression for the Laplace transform of a number of arrival distributions. In fact, we can compute (4.15) in closed form when $R$ is a mixture of polynomials and exponentials. In other cases, like the log-normal (Tellambura and Senaratne, 2010) or Pareto (Nadarajah and Kotz, 2006), there exist good approximations. In all these cases, solving (4.14) with a root-finding algorithm is quite fast with typical values of the order of 1 ms.

Things are much more difficult if there is no exact or approximate value for $\mathcal{F}_R(s)$. We still need to use a root-finding algorithm but in this case, we must also compute the Laplace transform numerically every time the solution procedure needs a value of $\mathcal{F}_R$ for some value of the argument $\tau$. This requires the numerical integration of $F_R$ in (4.15) every time we try a new value of $\tau$, which can lead to large computation times. For instance, in the case of the Gamma distribution, a solution using (4.53) will take less than 1 ms while the computation time when evaluating the Laplace transform by numerical integration would take nearly 300 ms.

**Computing the Jitter**

Once the value of $\eta$ has been computed, the calculation of the jitter reduces to the evaluation of the integral (4.15). The computational effort is similar to what is discussed in section 4.5. In some cases, the integral can be computed analytically, in which case the computation time is negligible. In other, more difficult cases, one has to do a single numerical integration and the computation times are of the same magnitude as the ones reported in Table 4.2 in section 4.8.2.

### 4.8.2 Limit Cases

We now present some detailed results for the computation of the bounds for very general queues in the two limit cases (4.29) and (4.33). They give some insight on the computational load required by single and double integration. We also give some analytic expressions for the jitter when they are available. First we recall some useful relations. Let $X$ and $Y$ be two independent random variables with probability density functions $f_X(x)$ and $f_Y(y)$ and define $Z$ as

$$Z = X - Y. \tag{4.55}$$

The pdf of $Z$ is the cross-correlation of $X$ and $Y$

$$\begin{aligned} f_Z(z) &= \int_{\Re} f_X(z+y) f_Y(y) dy \\ &= \int_{\Re} f_X(x) f_Y(x-z) dx \end{aligned} \tag{4.56}$$

where we assume that a distribution $f_X(x) = 0$ for $x$ outside its support. We are interested in the pdf $f_{|Z|}$ of $|Z|$ which is given by

$$\begin{aligned} f_{|Z|}(z) &= f_Z(z) + f_Z(-z) \\ &= f_Z^+(z) + f_Z^-(z) \end{aligned} \tag{4.57}$$

where $f^+$ and $f^-$ are called the positive and negative parts of $f_Z$ in the following. Replacing in (4.56), we get

$$f_{|Z|}(z) = \int_{\Re} f_X(z+y) f_Y(y) dy + \int_{\Re} f_X(y-z) f_Y(y) dy. \tag{4.58}$$

If $f_X$ and $f_Y$ are defined for $x, y \geq 0$, as will be the case here, we get

$$f_{|Z|}(z) = f_Z^+(z) + f_Z^-(z)$$

$$
\begin{aligned}
&= \int_0^\infty f_X(z+y) f_Y(y) dy + \\
&\quad \int_z^\infty f_X(y-z) f_Y(y) dy \\
&= \int_0^\infty f_X(z+y) f_Y(y) dy + \\
&\quad \int_0^\infty f_X(y) f_Y(z+y) dy
\end{aligned} \tag{4.59}
$$

for $0 \leq z < \infty$. If $X = Y$, this reduces to

$$f_{|Z|}(z) = 2 \int_0^\infty f_X(z+y) f_Y(y) dy. \tag{4.60}$$

Finally, we get the average as

$$E[|Z|] = \int_0^\infty z f_{|Z|}(z) dz. \tag{4.61}$$

We can get a simplified form for $J$ in the important case of G/D/1 queues where $f_S$ is deterministic of the form

$$f_S(y) = \delta(y - \overline{S}). \tag{4.62}$$

Replacing this in (4.59), we get

$$
\begin{aligned}
E\left[|Z|\right] &= \int_0^\infty z f_R(z+\overline{S}) + \int_0^\infty z f_R(\overline{S}-z) \\
&= \int_0^\infty z f_R(z+\overline{S}) + \int_0^{\overline{S}} z f_R(\overline{S}-z)
\end{aligned} \tag{4.63}
$$

since $f_R$ has positive support only.

Altogether, the complexity of the computation depends on whether we can get an analytic expression for (4.59) and (4.61), which in turn depends on the particular form of the density functions $f_R$ and $f_S$. We present computational results for three cases depending on which expression can be evaluated analytically : Both (4.59) and (4.61), only (4.59) or neither of them.

### Analytic Expressions for $J$

In some cases, we can find an explicit expression both for the probability density function (4.59) and for the average (4.61). This yields an analytic formula for the jitter and the computation time is negligible.

**G/M/1 at $\rho = 0$ :** A simple case where this happens is for the G/M/1 queue. Here, the limit (4.29) at $\rho \approx 0$ is given by the difference of two identical independent exponential variables with parameter $\mu$. This has a symmetric Laplace distribution centered at the origin and the distribution of the absolute value will be an exponential with parameter $\mu$. We have the same situation for the limit $\rho \approx 1$ where the two exponential variables $R$ and $S$ are exponentials with parameters $\lambda$ and $\mu$ where $\lambda \to \mu$. In both the low and high limits

$$J = \frac{1}{\mu} \tag{4.64}$$

as we have already seen in section 4.4.2.

**M/D/1 at $\rho = 1$ :** We can get an exact result for the M/D/1 queue at $\rho \approx 1$ by replacing $f_R(x) = \lambda \exp(-\lambda x)$ into (4.63). We get

$$\begin{aligned} J &= \lambda \int_0^\infty z e^{-(z+\overline{S})} + \int_0^{\overline{S}} z e^{-(\overline{S}-z)} \\ &= \frac{\lambda \overline{S} - 1}{\lambda} + \frac{2e^{-\lambda \overline{S}}}{\lambda}, \end{aligned} \tag{4.65}$$

$$\lim_{\lambda \overline{S} \to 1} J = \frac{2}{\lambda e}. \tag{4.66}$$

**G/Gm/1 at $\rho = 0$ :** Another case where we can get a complete analytic formula at $\rho \approx 0$ is the G/Gm/1 queue. We can compute the integral (4.59) and get

$$\begin{aligned} f_{|Z|}(z) = &\frac{1}{\Gamma(k)^2 \theta^{2k}} \\ &\Big[ e^{-z/\theta} \int_0^\infty (z+x)^{k-1} x^{k-1} e^{-2x/\theta} dx + \\ &e^{z/\theta} \int_z^\infty (x-z)^{k-1} x^{k-1} e^{-2x/\theta} dx \Big] \end{aligned} \tag{4.67}$$

which evaluates to (Gradshteyn and Ryzhik, 2007)

$$\begin{aligned} f_{|Z|}(z) = &\frac{z^{k-1/2}}{2^{k-1/2} \Gamma(k)^2 \theta^{k+1/2}} \times \\ &\left[ \frac{\sqrt{\pi} e^{z/\theta}}{\sin(k\pi)\Gamma(1-k)} + \frac{\Gamma(k) e^{-z/\theta}}{\sqrt{\pi}} \right] K_{k-1/2}(z/\theta) \end{aligned} \tag{4.68}$$

where $K_\nu(x)$ is the modified Bessel function of the second kind. Replacing (4.68) in (4.61), we get

$$J = \frac{2\theta\Gamma(1/2+k)}{\sqrt{\pi}\Gamma(k)} \quad \forall k > 0. \tag{4.69}$$

**Gm/D/1 at $\rho = 1$ :** We have a similar result for the Gm/D/1 queue at $\rho \approx 1$. When the arrival distribution is $Gm(k,\theta)$, we can evaluate $J$ using the two terms of (4.63). The positive part is given by

$$\int_0^\infty z f_R(z+\overline{S}) = \frac{1}{\Gamma(k)\theta}\int_0^\infty z\left(\frac{\overline{S}+z}{\theta}\right)^{k-1} e^{-(\overline{S}+z)/\theta}\,dz \tag{4.70}$$

Setting $y = (\overline{S}+z)/\theta$, this is equivalent to

$$\begin{aligned}\int_0^\infty z f_R(z+\overline{S}) &= \frac{1}{\Gamma(k)\theta}\int_0^\infty \left(\theta y - \overline{S}\right) y^{k-1} e^{-y}\,\theta dy \\ &= \frac{1}{\Gamma(k)}\left[\theta\Gamma\left(k+1,\frac{\overline{S}}{\theta}\right) - \overline{S}\Gamma\left(k,\frac{\overline{S}}{\theta}\right)\right].\end{aligned} \tag{4.71}$$

For the negative part, we get

$$\int_0^{\overline{S}} z f_R(\overline{S}-z) = \int_0^{\overline{S}} z\left(\frac{\overline{S}-z}{\theta}\right)^{k-1} e^{-(\overline{S}-z)/\theta}\,dz \tag{4.72}$$

and using the transformation $y = (\overline{S}-z)/\theta$, we get

$$\begin{aligned}\int_0^{\overline{S}} z f_R(\overline{S}-z) &= \frac{1}{\Gamma(k)\theta}\int_0^{\overline{S}/\theta}\left(\overline{S}-\theta y\right) y^{k-1} e^{-y}\,dy \\ &= \frac{1}{\Gamma(k)}\left[\overline{S}\gamma\left(k,\frac{\overline{S}}{\theta}\right) - \theta\gamma\left(k+1,\frac{\overline{S}}{\theta}\right)\right]\end{aligned} \tag{4.73}$$

where $\gamma(k,x)$ and $\Gamma(k,x)$ are the lower and upper incomplete gamma functions. These cases are summarized in Table 4.1.

Table 4.1 Analytic expression

| Queue | $\rho$ | Calculation |
|---|---|---|
| G/M/1 | 0 | (4.64) |
| M/D/1 | 1 | (4.65) |
| G/Gm/1 | 0 | (4.69) |
| Gm/D/1 | 1 | (4.71–4.73) |

**Analytic Expression for $f_{|Z|}$ Only**

More difficult cases happen when we can get an analytic expression for the distribution of the absolute value from (4.59) but where it is not possible to compute the average (4.61) analytically and we need to compute the integral (4.61) numerically. This is also the case for the G/D/1 at $\rho = 1$ where we cannot compute (4.63) analytically.

**M/Nt/1 at $\rho = 1$ :** The first example is the M/Nt/1 queue where the service time $S$ follows a normal distribution $Nt(x; \mu, \sigma)$ truncated at 0 with location $\mu$ and scale $\sigma$. The pdf of the service time random variable $S$ is then given by

$$Nt(x; \mu, \sigma) = \frac{1}{\sigma \Phi\left(\mu/\sigma\right)} \phi\left(\frac{x-\mu}{\sigma}\right) \tag{4.74}$$

where $\phi(\cdot)$ is the standard normal distribution and $\Phi(\cdot)$ its corresponding cdf (See Appendix.4.11.4). We get for the two terms of (4.59)

$$\begin{aligned} f_Z^+(z) = \frac{\lambda}{\Phi\left(\mu/\sigma\right)} \exp\left(\frac{\lambda^2\sigma^2}{2} - \lambda(\mu + z)\right) \times \\ \Phi\left(\frac{\mu - \lambda\sigma^2}{\sigma}\right) \end{aligned} \tag{4.75}$$

$$\begin{aligned} f_Z^-(z) = \frac{\lambda}{\Phi\left(\mu/\sigma\right)} \exp\left(\frac{\lambda^2\sigma^2}{2} - \lambda(\mu - z)\right) \times \\ \Phi\left(\frac{\mu - z - \lambda\sigma^2}{\sigma}\right) \end{aligned} \tag{4.76}$$

but it is not possible to get a closed form for (4.61) from this and we need to compute the integral numerically.

**LogN/D/1 at $\rho = 1$ :** We get the same case for the LogN/D/1 queue for $\rho \to 1$ where $R$ follows a log-normal distribution $LogN(x; , \mu, \sigma)$ and $S$ is deterministic. We cannot eva-

luate (4.63) analytically so we need to compute the integral (4.63) numerically.

$$f_Z^+ = e^{\left(\frac{1}{2}\sigma^2+\mu\right)}\Phi\left(\frac{\sigma^2+\mu-\log(\overline{S})}{\sigma}\right) \tag{4.77}$$

$$f_Z^- = e^{\left(\frac{1}{2}\sigma^2+\mu\right)}\Phi\left(-\frac{\sigma^2+\mu-\log(\overline{S})}{\sigma}\right) \tag{4.78}$$

We can get the distribution of $J$ by (4.57) from the sum of the two terms (4.77–4.78) but here again it is not possible to compute the mean value analytically.

**G/Nt/1 at $\rho = 0$ :** The pdf of the service time is that of a truncated normal random variable $S$. We can evaluate the two parts of (4.59) analytically and we get

$$\int_0^\infty f_S(z+x)f_S(x)dx = \frac{1}{2\sigma\sqrt{\pi}\left[\Phi\left(\frac{\mu}{\sigma}\right)\right]^2}\exp\left(-\frac{z^2}{4\sigma^2}\right)\Phi\left(\frac{2\mu-z}{\sigma\sqrt{2}}\right) \tag{4.79}$$

$$\int_{-z}^\infty f_S(z+x)f_S(x)dx = \frac{1}{2\sigma\sqrt{\pi}\left[\Phi\left(\frac{\mu}{\sigma}\right)\right]^2}\exp\left(-\frac{z^2}{4\sigma^2}\right)\Phi\left(\frac{2\mu+z}{\sigma\sqrt{2}}\right) \tag{4.80}$$

We can compute the pdf of $|Z|$ by replacing (4.79) and (4.80) in (4.57) which gives

$$f_{|Z|}(z) = \frac{1}{\sigma\sqrt{\pi}\left[\Phi\left(\frac{\mu}{\sigma}\right)\right]^2}\exp\left(-\frac{z^2}{4\sigma^2}\right)\Phi\left(\frac{2\mu+z}{\sigma\sqrt{2}}\right). \tag{4.81}$$

In all these cases, Eq. (4.61) has to be evaluated numerically with a significant computation time. Some values for the computation times are presented in Table 4.2. We fix the mean of $R$ to 0 or 1 as the case may be and measure the cpu time for different values of the standard deviation $s$. The worst case is for the M/Nt/1 queue at $\rho = 1$ which needs about 200 ms to compute. We see that in most cases, the integration is quite fast so that it would be possible to use this within some iterative procedure.

### Double Numerical Integration

An even more difficult situation arises when it is not possible to evaluate (4.59) analytically. In this case, we evaluate (4.61) by computing numerically a double integral : one for the distribution function and one for the average. We present in Table 4.3 some numerical results

Table 4.2 Cpu times for single integration

| Queue | $\rho$ | $s$ | cpu sec |
|---|---|---|---|
| G/Nt/1 | 0 | 0.15 | 0.037 |
| G/Nt/1 | 0 | 0.25 | 0.036 |
| G/Nt/1 | 0 | 0.35 | 0.036 |
| M/Nt/1 | 1 | 0.15 | 0.20 |
| M/Nt/1 | 1 | 0.25 | 0.25 |
| M/Nt/1 | 1 | 0.35 | 0.21 |
| Gm/D/1 | 1 | 0.25 | 0.002 |
| Gm/D/1 | 1 | 1.5 | 0.006 |
| Gm/D/1 | 1 | 3.5 | 0.008 |
| Logn/D/1 | 1 | 0.5 | 0.004 |
| Logn/D/1 | 1 | 1. | 0.005 |
| Logn/D/1 | 1 | 2. | 0.008 |

for the M/Nt/1 queue in both limit cases. We use the same computation tool as in section 4.8.2 but use the `dblquad` function for double integration.

We repeat the calculation for the G/Nt/1 queue at $\rho = 0$ that is shown in Table 4.2 to estimate the difference in cpu time when using double as opposed to single integration. Typical computation times by double integration are about 3 orders of magnitude larger than for single integration, around 10 seconds. It would be difficult to use double integration iteratively or in real time. Still, we could compute these values off-line and use them as input to the approximation algorithms for the estimation of jitter as a function of traffic when $f_T$ is not known.

Table 4.3 Cpu times for double integration

| Queue | $\rho$ | $s$ | cpu sec |
|---|---|---|---|
| G/Nt/1 | 0 | 0.15 | 11.8 |
| G/Nt/1 | 0 | 0.25 | 13.0 |
| G/Nt/1 | 0 | 0.35 | 12.4 |
| M/Nt/1 | 1 | 0.15 | 15.5 |
| M/Nt/1 | 1 | 0.25 | 10.2 |
| M/Nt/1 | 1 | 0.35 | 10.3 |

### 4.8.3 G/D/1

For the three-segment model, we need the three points $P_0$, $P_1$ and $P_2$. The first point $P_0$ is determined immediately, since for G/D/1 queues,

$$\lim_{\rho \to 0} J(\rho) = 0. \tag{4.82}$$

The time required to compute $P_2$ when $\rho \approx 1$ is equivalent to time evaluated in section 4.8.2. Finally, computing the intermediate point $P_1$ requires the solution of equation (4.39). It takes about 3 ms when the inter-arrival time $R$ follows a Gamma distribution and approximately the same for a Lognormal distribution. Altogether, the computation is dominated by the computation of $P_2$ and despite the fact that the approximation may not be very accurate in some cases , the simplicity and the computation speed could still make it very useful.

## 4.9 Conclusion

We proposed some techniques for the calculation of jitter for different types of FCFS queues with a single stream. First we recalled a general formula proposed previously and pointed out two problems : One is that it depends on the distribution of the sojourn time, which is often not known, and two, that it takes the form of a triple integral, which can be difficult to solve numerically.

We first proposed an exact model for G/M/1 queue since the sojourn time follows an exponential distribution whose parameter is the solution of a nonlinear equation involving the Laplace transform of the inter-arrival distribution. In the worst case, when the transform has to be computed numerically, solving this equation required in the order of a few hundred milliseconds. We then reduced the calculation of the triple integral to a single integral which can be computed numerically quickly. This approach is feasible because the sojourn time parameter can be evaluated offline outside the iteration procedure.

We then gave two expressions for a G/G/1 queue at low and high traffic that do not depend on the sojourn time. We also gave examples where it is possible to compute these values analytically or with a single numerical integration. More complex cases require a double integral which require about 10 to 20 seconds. These limit points are then used to construct approximate models for queues where the sojourn time distribution is not known.

Next, we showed that for the M/G/1 queue, a linear interpolation between the low and high limits is an excellent approximation of the actual jitter so that the computation time is essentially that of the limit points.

Finally, we presented a good approximation for the G/D/1 queue using a three-segment interpolation. The computational requirement is the solution of a nonlinear equation for the cdf of the arrival process. The results were compared with simulation results indicating a reasonably good accuracy with poorer results with processes with a high variance.

Our conclusion is that we can use these expressions to evaluate quickly the jitter in a queue with reasonable accuracy and for a number of important processes. The computation speed is small enough that we can use them to investigate the effect of jitter in network design and optimization models.

We also get a better understanding of the properties of jitter as compared with delay. We find that jitter does not go always to zero at low load and does not go to infinity at high load where it can sometimes decrease with increasing load. In fact, the values seldom exceed the holding time and when they do, only by a small value. This seems to happen when processes have a large variance, something that is not totally unexpected. An even more counter-intuitive result is that in some cases, jitter can actually *decrease* when the load increases. These properties seem to hold for a range of arrival and service processes, which lead us to believe that they may in fact be quite general.

## 4.10 Acknowledgements

This work was supported by Grant No. 121949-2012 from the Natural Sciences and Engineering Research Council of Canada.

## 4.11 Notation

For the sake of completeness, we recall the definitions of the distributions used in the paper.

### 4.11.1 Gamma

The cdf is the regularized gamma function

$$F_R(x,k,\theta) = \frac{1}{\Gamma(k)} \int_0^{x/\theta} t^{k-1} e^{-t} dt. \tag{4.83}$$

The parameters are related to the mean $m$ and variance $v = s^2$ of the distribution by

$$m = k\theta \tag{4.84}$$

$$v = k\theta^2 \tag{4.85}$$

which we can solve to get

$$\theta = \frac{v}{m} \tag{4.86}$$

$$k = \frac{m^2}{v}. \tag{4.87}$$

### 4.11.2 Pareto

The Pareto Type I $P(x; x_m, \alpha)$ with scale $x_m$ and shape $\alpha$ has a density function given by

$$f_R(x) = \begin{cases} \alpha \dfrac{x_m^\alpha}{x^{\alpha+1}} & \text{if } x \geq x_m \\ 0 & \text{otherwise.} \end{cases} \tag{4.88}$$

The Pareto parameters $\alpha$ and $x_m$ are related to the mean $m$ and variance $v = s^2$ by

$$m = \begin{cases} x_m \dfrac{\alpha}{\alpha - 1} & \text{if } \alpha > 1 \\ \infty & \text{if } \alpha \leq 1 \end{cases} \tag{4.89}$$

$$v = \begin{cases} x_m^2 \dfrac{\alpha}{(\alpha-1)^2(\alpha-2)} & \text{if } \alpha > 2 \\ \infty & \text{if } \alpha \leq 2. \end{cases} \tag{4.90}$$

For $\alpha > 2$, we can solve (4.89–4.90) to get

$$\alpha = 1 + \sqrt{1 + (m^2/v)} \tag{4.91}$$

$$x_m = m \frac{\alpha - 1}{\alpha}. \tag{4.92}$$

### 4.11.3 Normal

We denote the pdf of the standard normal distribution

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \exp(-x^2/2) \tag{4.93}$$

and the corresponding cdf

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp(-t^2/2) dt. \tag{4.94}$$

The error function is given by

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{4.95}$$

and we have the relation

$$\Phi(x) = \frac{1}{2} + \frac{1}{2} \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right). \tag{4.96}$$

The pdf of a normally distributed random variables $X$ with location $\mu$ and scale $\sigma$ is given by

$$f_X(x; \mu, \sigma) = \frac{1}{\sigma} \phi\left(\frac{x-\mu}{\sigma}\right) \tag{4.97}$$

and the corresponding cdf by

$$F_X(x; \mu, \sigma) = \Phi\left(\frac{x-\mu}{\sigma}\right). \tag{4.98}$$

### 4.11.4 Truncated Normal

The pdf of the truncated normal is that of a normal random variable $X$ with location and scale parameters $\mu$ and $\sigma$ and truncated to the interval $[0, \infty]$, is given by

$$f_X(x; \mu, \sigma) = Nt(x; m, s) = \frac{1}{\sigma \Phi\left(\mu/\sigma\right)} \phi\left(\frac{x-\mu}{\sigma}\right) \tag{4.99}$$

which is simply a gaussian distribution with support $[0, \infty]$ with the appropriate normalization. The mean $m$ and variance $v = s^2$ are given by

$$m = \mu + \sigma A \tag{4.100}$$

$$v = \sigma^2 (1 - B) \tag{4.101}$$

where we have defined

$$A = \frac{\phi\left(\frac{\mu}{\sigma}\right)}{\Phi\left(\frac{\mu}{\sigma}\right)} \tag{4.102}$$

$$B = A\left(A - \frac{\mu}{\sigma}\right). \tag{4.103}$$

Note that the parameters $\mu$ and $\sigma$ need not exist for an arbitrary choice of $m$ and $s$.

### 4.11.5 Log-Normal

This is the distribution of a random variable $X$ such that $\log X$ is normally distributed. The pdf for location $\mu$ and scale $\sigma$ is given by

$$f_X(x;\mu,\sigma) = \frac{1}{x\sqrt{2\pi}\sigma}\ \exp\left(-\frac{(\ln x-\mu)^2}{2\sigma^2}\right) \tag{4.104}$$

The mean $m$ and variance $v = s^2$ are given by

$$m = e^{\mu+\sigma^2/2} \tag{4.105}$$

$$v = (e^{\sigma^2}-1)e^{2\mu+\sigma^2} \tag{4.106}$$

which can be inverted to give

$$\mu = \ln\left(\frac{m^2}{\sqrt{v+m^2}}\right) \tag{4.107}$$

$$\sigma = \sqrt{\ln\left(1+\frac{v}{m^2}\right)} \tag{4.108}$$

# CHAPITRE 5 ARTICLE 2 : ON THE RELATIONSHIP BETWEEN PACKET JITTER AND BUFFER LOSS PROBABILITIES

Recopié avec permission : H. Dbira, A. Girard et B. Sansò, *On the Relationship between Packet Jitter and Buffer Loss Probabilities*, accepté à la $17^{ième}$ édition de la conférence Networks, 2016 - IEEE.

**Abstract**

Controlling the jitter buffer at the receiver side is a key component in multimedia and real-time services. A jitter buffer is used to control the packet rate and to compensate for the delay variation, commonly called jitter, by the tradeoff between delay and loss. In this paper, we present simulation results for the relation between the over- and underflow probabilities and the jitter. We then show how we can control a jitter buffer based on an estimation of the variance of the arrival process and a fast computation model for the jitter in this buffer.

## 5.1 Introduction

The fast growth of the Internet is currently driven by the rapid spread of multimedia streaming applications and real-time services such as television over IP, video conferencing, video on demand, telemedicine, etc. These services require a strict quality of service (QoS) expressed as bounds on parameters like packet loss, delay and delay variation, also known as *jitter*.

In particular, multimedia traffic is very sensitive to jitter and an important solution currently used to control jitter is to store the received packets in a buffer, commonly called the *playout* or *jitter buffer*, until they can be played back. In that context, a service interruption happens when this client buffer becomes empty because the network cannot deliver packets fast enough for some time. This is known as a buffer underflow. Another cause of Quality of Experience (QoE) degradation is when the buffer is full and packets are dropped, which leads to picture degradation such as pixelation. This is called *buffer overflow* and is generally considered less annoying than an actual freeze of the media stream. In the following, we will use the termp *failure* to denote either of these conditions. Thus, signal loss caused by either underflow and overflow is an important performance metric for both the jitter buffer control and the QoE estimation (ParandehGheibi et al., 2011).

There are two general classes of algorithms currently used to control playback buffers in IP networks. One is *reservation-based*, where resources are dynamically allocated in advance along the path across the network carrying the service. This does not scale well and most techniques use *feedback-based* control. With these techniques, the receiver monitors the buffer for impending overflow or underflow. When failure is detected, the controller either modifies the playback rate or signals the source that something is wrong. The source will then change its sending parameters, e.g., the rate, the encoding, etc.

In this paper, we are interested in the receiver-side detection techniques. Most of these monitor the buffer occupancy directly and use a threshold policy to decide when some remedial action is needed. Many control techniques are proposed in the literature (Xu et al., 2014; Seo et al., 2008) based on minimizing the over- and underflow probability $P_o$ and $P_u$.

There also have been proposals to use a different metric to monitor buffer occupation, in particular the packet delay variation, or jitter. While there is a good reason why we might want to use jitter as core control variable, as explained later on, this raises two questions which are at the core of this paper. They are formulated in Sections 5.1.1 and 5.1.2 below.

### 5.1.1 Extension of the Jitter Model

There has been much work to characterize playout buffer for Asynchronous Transfer Mode (ATM) networks during the 1990s. The work of (Omnes et al., 1998) considers only an upper bound for buffer overflow ratio to size the playout buffer. Authors in (Kelly and Key, 1994) address the effects on service delivery performance of failure in playout buffers. They propose to compensate jitter, *i.e., cell delay variation*, by an intial buffer prefetching which is computed based on the difference between maximum and minimum cell end-to-end delays.

Much of this work is tailored to the specific architecture of ATM networks where all cells have the same size and are all 48 byte long. Also, a basic assumption is that it is enough to guarantee a low enough loss rate, of the order of $10^{-8}$, to insure that all other relevant QoS requirements are met. This is quite different from IP networks, where packets have different sizes and are much larger, typically up to about 1500 bytes. Also, the loss requirements may be much larger than the ATM, typically of the order of 1%. The same goes for average delay, which can be as large as about 100 ms without significant impairment. For all these reasons, we have found that the ATM work is not readily usable for IP network.

The techniques that currently use jitter as an estimator for buffer congestion or underflow use actual measurements of network delay. These measurements (Ramjee et al., 1994; Narbutt and Murphy, 2004) are usually performed on packet samples, use a linear recursive tech-

niques and need to estimate packet round-trip time or to consult header timestamps. These measurements may take some time and using one half of the RTT is only an approximation of the transit delay since delay need not be the same on both directions of a path or the return path may be different from the path used to carry the stream.

These shortcomings could be avoided if the receiver could actually *compute* the effect of some parameter change on the jitter. It could then see in advance what would be the effect of a change of parameters by the sender and make a recommendation accordingly. Such an approach has become possible due to some recent results that provide fast and accurate computation models for estimating jitter (Dahmouni et al., 2012a; Dbira et al., 2014) for some general types of queues. The problem is that the models have been derived for queues with infinite buffers while jitter buffers are relatively small to avoid long delays. While there is a rich literature on queues with finite buffers, we have not been able to find any results relevant to computing the jitter.

Therefore, the first objective of this paper is to determine the accuracy of the infinite buffer model when used with a finite-buffer queue.

### 5.1.2 Jitter as a Proxy

The second question of interest is to what extent jitter can be used as a tool to control the buffer. It is not unreasonable to think that a stream with low jitter, and thus more regular traffic, would have lower failure probability than one with a larger jitter as suggested in (Demichelis and Chimento, 2002; Clark and Claise, 2011; Clark and Wu, 2012). A recent survey (Morton and Claise, 2009) discusses different uses of jitter as a control variable for the playout buffer. The authors claim that it would be a good measure to predict link congestion and network stability since it is a measure of the ability to preserve packet spacing. They also note that it is possible to evaluate QoS through this metric when there are frequent path changes. Authors in (Angrisani et al., 2013) have shown that there is a definite correlation between the delay variation as defined in (Demichelis and Chimento, 2002) and video impairments. Jitter is also used in (Li et al., 2012) to compute upper and lower thresholds for the occupancy of a jitter buffer beyond which some remedial action is needed.

Still, some work indicates that it may not be so in some situations. A case in point is the statement in (Clark, 2003) to the effect that "*[jitter] is not effective for larger congestion related jitter events*" . It is also stated in (Morton and Claise, 2009) that there is no guarantee that jitter measurements would be useful either for sizing jitter buffers or as a control variable. This would suggest that jitter may not be very tightly related to the actual QoS, at least in some cases.

In fact, we do know there are cases where the connection between jitter and buffer management simply does not exist. It is known (Dahmouni et al., 2012a) that for a M/M/1 queue, the jitter is equal to the mean service time and thus independent of any traffic variation. Similarly, in a D/D/1 queue, where the jitter is always 0, the buffer could be always empty or always full depending on the arrival and service rates. In fact, there will eventually be a failure for any buffer where the arrival rate is either larger or less than the service rate, irrespective of the jitter. For cases like these, it is not possible to use jitter as a tool for managing the buffer. However, our intuition is that for more general distributions, that correlation does exist, can be evaluated and can be used for jitter control.

Thus, the second objective of this work is to test this intuition experimentally by studying the correlation between the performance of playout buffers as measured by $P_u$ and $P_o$ and jitter as defined in (Demichelis and Chimento, 2002).

Our final objective is to use that correlation to propose a fast algorithm to compute $P_u$ and $P_o$ based on jitter monitoring.

### 5.1.3 Paper Structure

First, in section 5.2, we recall some results (Dbira et al., 2014) that are used later. Next, the first set of results in section 5.3 shows that the infinite buffer model can be quite accurate even for relatively small buffers. The use of jitter as a proxy for failure is described in section 5.4 where we present some simulation results showing that jitter measured in the playout buffer is well correlated to the underflow and overflow probability for realistic non-Poisson arrival processes. We then explain how to use this to compute the buffer probabilities given an estimation of the variance of the arrival process. An algorithm to this effect is introduced in this section. Finally, concluding remarks are discussed in section 5.5.

## 5.2 Infinite Buffer Model

### 5.2.1 Jitter and Mean Delay Variation

*Delay variation*, *network jitter*, or simply *jitter* is a measure of the variation of delay in a sequence of packets over time. This is generally due to packet queuing within the network and is one of the most important parameters characterizing network transmission and QoS.

The formal definition of jitter differs according to different standards organizations. The International Telecommunication Union-Telecommunication Standardization Sector (ITU-T) defines jitter as the variation of delay from some reference value $a_i$. It is the difference

between the one-way delay of a packet $i$, $T_i$, and some reference delay $a_i$ like the minimum or mean delay (itu, 2011; Clark, 2003).

$$J = E\left[|T_i - a_i|\right]. \tag{5.1}$$

With this definition, it is possible to quickly detect an increase in packet delay and for this reason, this definition has been used for computing the jitter buffer size (Morton and Claise, 2009) to prevent buffer overflow.

Let $T_i$ and $T_{i+1}$ be the transit time of consecutive packets. The Internet Engineering Task Force (IETF) defines the delay variation as the difference $T_{i+1} - T_i$ between two measurement points (Demichelis and Chimento, 2002; Angrisani et al., 2013). The jitter is defined as the expected absolute value of this random variable

$$J = E\left[|T_i - T_{i+1}|\right] \tag{5.2}$$

and for referring the jitter in this IETF memos they use the term Mean Packet-to-Packet Delay Variation(MPPDV).

Based on recent work (Dahmouni et al., 2012a; Dbira et al., 2014), we now have accurate and fast computation models for estimating $J$ given by (5.2). For this reason, we want to study how we can use jitter to estimate the loss probabilities in jitter buffers.

### 5.2.2 Jitter Buffer Model

A multimedia transmission system over an IP network is illustrated in figure 5.1. The quality impairment caused by the jitter is compensated at the client side by a jitter buffer. The packet traffic is read off this buffer and sent afterwards to the decoder. In this paper, we

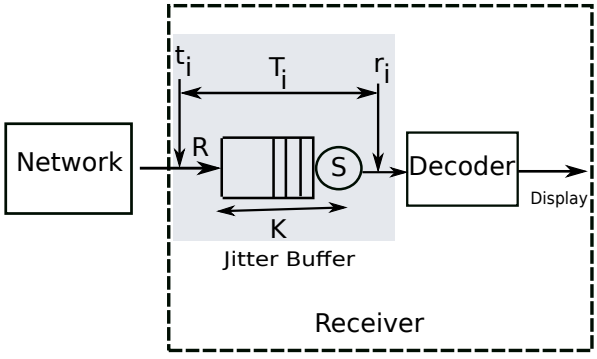



Figure 5.1 Multimedia Transmission System

consider only the jitter buffer of a particular application as shown by the shaded area of the figure.

We assume that the interarrival distribution of the packets arriving at the buffer follows a load-independent general distribution and that the packet service time is constant. This is not unrealistic for video traffic since for a standard Mpeg4 codec, for instance, the Intra-coded frames, which are coded without reference to other frames, are generally much larger than the standard 1500-byte maximum transmission unit so that most of the packets are of this length. Also, there is some evidence (Fraleigh et al., 2003) that a large portion of Internet traffic is made up of packets of particular sizes with only a few specific values instead of a continuous distribution. We also assume that there is room for $K-1$ packets in the buffer and a First-Come First-Served (FCFS) service discipline is used so that the jitter buffer can be modeled as a G/D/1/K queue.

Recall that we are considering the playout buffer for a single application. This means that the packet generation rate at the source codec must be equal to that arriving at the destination server, at least in the long term. Otherwise, some failure is guaranteed to occur at some time. This is why we assume in what follows that the service rate of the queue $\mu$ is equal to the traffic arrival rate $\lambda$, so that the traffic load $\rho = \lambda/\mu = 1$.

From this model we can define the jitter buffer failure probabilities. An overflow occurs when we have a full buffer upon the arrival of a new packet, and an underflow when the buffer is empty when the next packet is needed for decoding.

### 5.3 Accuracy for Finite Buffer

Given that computing the jitter with the infinite-buffer model of (Dbira et al., 2014), is relatively simple, we would like to use it for computing the jitter for a finite-size playout buffer. It is clear that this will be a good approximation when $K$ is large so that the real question is the accuracy when $K$ becomes small.

We use simulation to compare the jitter for a G/D/1/K queue with results computed by equation ((51)) from (Dbira et al., 2014) for an infinite queue. We use Network Simulator (NS-2) for simulating a G/D/1/K queue. A traffic source generates packets with constant packet size and the packet interarrival time follows a continuous stochastic process. Note that this source is used for simulating the arrival process at the receiver-side only and has nothing to do with the network transmission process. For all simulations, the service rate of these queue $\mu$ is equal to the arrival rate $\lambda = 1/E[R_i]$ in order to get $\rho = 1$ as required for the model described above.

We present results for a gamma interarrival process in figure 5.2 and for a lognormal in figure 5.3. The results produced by the infinite-buffer model, labeled `Analytic`, are computed for $\lambda \simeq \mu = 1$. The curves are produced as a function of the standard deviation of the interarrival distribution. The same conditions are applied for simulation results, labeled `Sim`, for different values of the buffer capacity $K - 1$.

To summarize, we see from figures 5.2 and 5.3 that the G/D/1/$\infty$ model becomes quite close to the curve for a finite capacity of $K \approx 120$. Moreover, we see that the accuracy is quite good for values of $K$ as low as 20. For smaller values, we also note that the infinite-buffer values are an upper bound on the actual values so that using them provides a conservative estimate of the jitter. In fact, in simulations and experiments studies (Li, 2013; ParandehGheibi et al., 2011) playout buffer size is usually large and can reach in some cases 1000 packets. For this reason, we consider that the fast jitter calculation is a reasonable estimate of the jitter and this is what we will be using in the following.

## 5.4 Estimating the Probabilities

In this section, we look into the relationship between the jitter $J$ and the buffer failure probabilities and present a technique to estimate $P_o$ and $P_u$ based on the calculation of the jitter $J$ as described in section 5.2. In what follows, $P_u$ is defined as the ratio of the server idle periods to the total simulation time and the $P_o$ is the ratio of dropped packets to the total number of generated packets, in accordance to section 5.2.2.

### 5.4.1 Relation between $J$ and $P$

We now present some examples of simulation results indicating that there is in fact a rather strong relation between jitter and probabilities for realistic arrival processes. Note that all

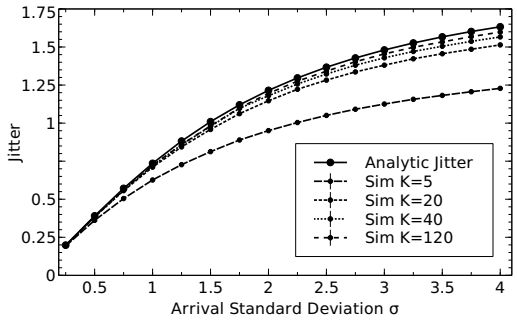



Figure 5.2 Mean Delay Variation : Gm/D/1 Vs Gm/D/1/K

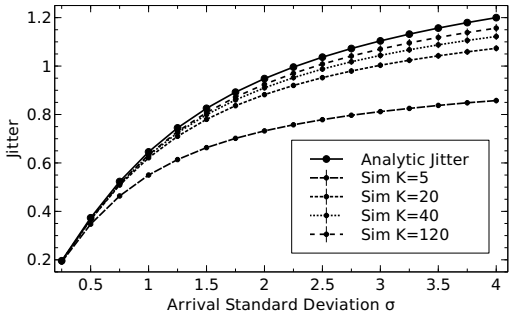



Figure 5.3 Mean Delay Variation : LogN/D/1 and LogN/D/1/K

measurements are made at the entry of the jitter buffer in the client equipement. First, we consider a Gm/D/1/K queue where the interarrival time follows a Gamma distribution with the two cases $K = 5$ and $K = 140$ with a fixed packet size of 1500 bytes. This queue operates at $\rho = 1$ with a single flow. Results are given in figures 5.4 and 5.5 and expressed in service time unit. In these figures, we plot the jitter and failure probabilities as a function of the standard deviation $\sigma$ of the inter-arrival time distribution. The vertical axis on the left refers to the jitter values and the right one is for loss and overflow probabilities. Jitter is computed using the formula given in (Dbira et al., 2014). The values of $P_o$ and $P_u$ are measured from a simulation and the 95% confidence interval is also shown in all figures. Other results are given in figures 5.6 and 5.7 when the interarrival time following the LogNormal distribution.

To summarize, we see that the jitter is correlated with loss probabilities. Although not linear, the relation is quite evident for small buffers and somewhat less clearly, also for large buffers. Results show then that notwithstanding the particular cases discussed in section 5.1, there is in fact a rather strong connection between jitter and the buffer failure probabilities. Based on this, we accept the notion that jitter can be used as a proxy for failure probability.

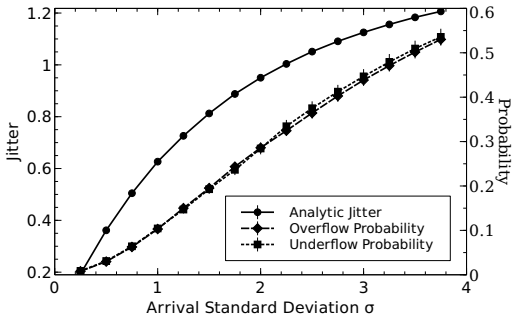



Figure 5.4 Gm/D/1/5 : Small Buffer

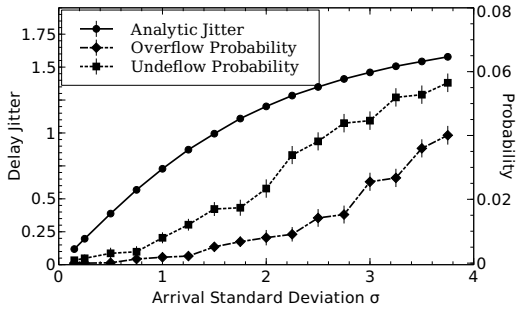



Figure 5.5 Gm/D/1/140-Large Buffer

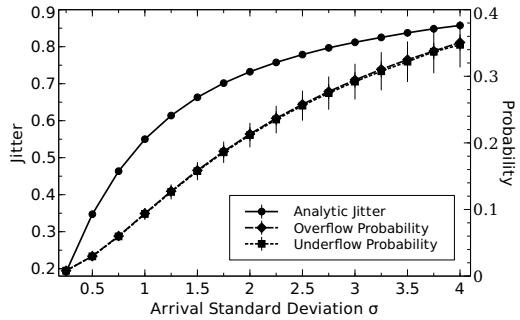



Figure 5.6 LogN/D/1/5-Small Buffer

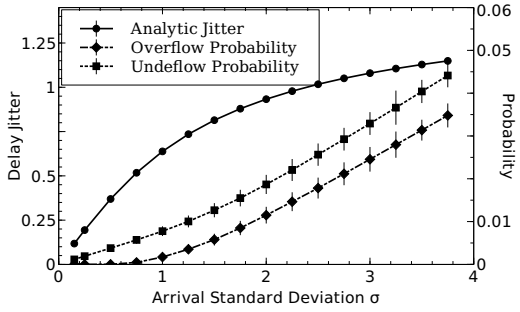



Figure 5.7 LogN/D/1/140-Large Buffer

### 5.4.2 Estimating the Arrival Distribution

Before we can use jitter to estimate failure probability, we still need to characterize the G/D/1 queue that we will be using. We have seen that the deterministic service time is a reasonable approximation so that we need only a reasonable estimate of the arrival process G. This process is determined by the delays exprerienced by the packets during their transmission in the network which is generally unknown. We try a number of distributions and in each case, we plot the failure probability against the jitter to see if the choice of distribution has a significant impact on the probability. We then choose a distribution that seems to be a worst-case scenario in the sense that it yields larger probabilities than the others. We then assume *for the purpose of calculating the probabilities* that this is the actual arrival distribution. Note that we are *not* making any claim that the distribution is in fact the real one. All we are saying is that it is good enough for the estimation.

We start by presenting the results of section 5.4.1 in the jitter-probability space for buffer sizes of $K = 5$ and $K = 140$. We use different distributions for G such as Gamma, LogNormal, $\beta$-Inverse and Hyper-Exponential (MMPP).

We plot $P_o$ as a function of the jitter in figure 5.8 and $P_u$ in 5.9 for $K = 5$. We can see from these two plots that for a small buffer, the actual distribution of the interarrival time has little effect on the relation betweed the jitter and the buffer failure probabilities. This is particularly interesting since some real-time jitter buffers are commonly minimized to keep the delay low for real-time applications.

The results are somewhat different for a large buffer $K = 140$ as can be seen in figures 5.10 and 5.11 where $P$ vs $J$ curves are a bit more distinct in some cases. Note however that the log-normal yields a value of the failure probability that is generally larger than that produced from the other distributions so that using this distribution would be a conservative estimate with respect to the actual distribution.

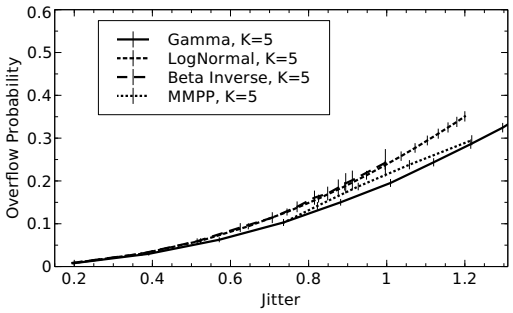



Figure 5.8 Jitter Vs Overflow Probability : Small Buffer

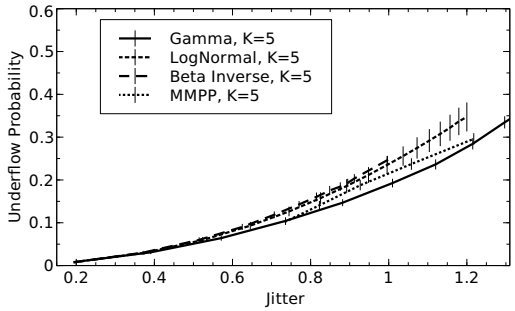



Figure 5.9 Jitter Vs Underflow Probability : Small Buffer

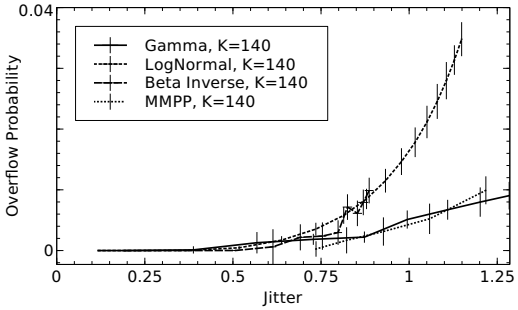



Figure 5.10 Jitter Vs Overflow Probability-Large Buffer

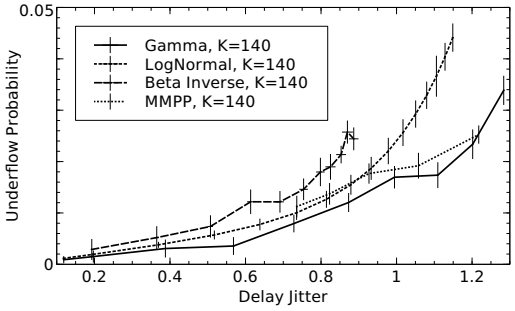



Figure 5.11 Jitter Vs Underflow Probability-Large Buffer

For this reason, we take the view that we can use a log-normal distribution for the arrival process and get either a good approximation for small buffer or a realistic upper bound for a large buffer.

### 5.4.3 Using $J$ to Estimate $P$

We can use these results to estimate the buffer probabilities if we measure the standard deviation $\sigma$ of the arrival process. Given an estimation of the variance, we can then compute the jitter using the technique described in (Dbira et al., 2014) and read the probability values off the curves. To simplify the discussion, we denote the relationship between the probabilities and the jitter by the notation $P_o = f_K(J)$ and $P_u = g_K(J)$. These curves can be computed offline and stored in the jitter buffer controller.

We propose a new algorithm for jitter buffer controller modules whose main function is estimating buffer loss probabilities based on jitter estimation. The method, given by algorithm 1, returns the buffer failure probabilities through an analytic method compatible with a G/D/1/K jitter buffer model. We present in the following the details of each step of about the algorithm.

Algorithm 1 Buffer Failure Probability Estimation

1: Initialization : $K$, $f_K(J)$ and $g_K(J)$ stored in controller memory
2: Measurements : $\sigma$ and $m$ if needed
3: Compute the parameters $\omega$, $s$ of the log-normal distribution
4: Compute the jitter
5: Calculate the buffer probabilities

**Initialization** The curves describing the failure probabilities as a function of $J$ are prestored in the controller. We also store other relevant parameters such as $K$ and, if available, the service rate. This would happen when we know what kind of codec is being fed by the buffer and what is the average rate needed.

**Measurements** We need to measure the standard deviation $\sigma$ of the packet interarrival time variable $R$. If the average interarrival time is not known, it can also be measured and in this case, this value is taken to be the average service time $\overline{S}$ as well to meet the condition $\rho = 1$. This could be performed at each time period. Let $A_T$ be the number of packets that

arrive in time period $T$, then

$$\overline{S} = m = \frac{1}{A_T} \sum_{i=1}^{A_T} R_i \tag{5.3}$$

$$\sigma = \sqrt{\frac{1}{A_T} \sum_{i=1}^{A_T} (R_i - m)^2} \tag{5.4}$$

**Compute the Parameters** From the mean and standard deviation of the arrival process measured in the previous setp, we compute the location parameter $\omega$ and scale parameter $s$ of the log-normal distribution.

$$\omega = \ln\left(\frac{m^2}{\sqrt{\sigma^2 + m^2}}\right) \quad s = \sqrt{\ln\left(1 + \frac{\sigma^2}{m^2}\right)} \tag{5.5}$$

**Calculating the Jitter** From these, we can compute the jitter using (Dbira et al., 2014) when $R$ follows a log-normal distribution

$$\begin{aligned} J = &\int_0^{\infty} z \left( \frac{1}{(z+\overline{S})\sqrt{2\pi}s} \exp\left( -\frac{\left(\ln(z+\overline{S}) - \omega\right)^2}{2s^2} \right) \right) dz \\ &+ \int_0^{\overline{S}} z \left( \frac{1}{(\overline{S}-z)\sqrt{2\pi}s} \exp\left( -\frac{\left(\ln(\overline{S}-z) - \omega\right)^2}{2s^2} \right) \right) dz \end{aligned} \tag{5.6}$$

**Computing the Buffer Probabilities** Given the value of $J$, we can finally read $P_o$ and $P_u$ from the stored curves for $f_K(J)$ and $g_K(J)$.

## 5.5 Conclusion

We found that we can approximate the jitter in a G/D/1/K queue by an infinite-buffer model for buffer sizes as low as 20, a value which is appropriate for a playout buffer. We have examined the relation between the jitter and the buffer probabilities in a G/D/1/K queue. We found that those parameters are reasonably well correlated, especially for small buffers. We then showed how this could be used to estimate quickly the buffer probabilities based on an estimation of the variance of the arrival process regardless the interarrival time distribution. Playout buffer failure probabilities estimation is very important for QoE evaluation and in another hand is requisite for buffer dimensioning.

# CHAPITRE 6 ARTICLE 3 : END-TO-END QUEUING MODEL EQUIVALENT FOR VIDEO APPLICATIONS

Recopié avec permission : H. Dbira, A. Girard et B. Sansò, *End-to-End Network Queuing Model Equivalent for Video Applications*, soumis au journal Compuer Networks, Elsevier.

**Abstract**

Network characterization and modelling is an important issue to understand and monitor IP network performance, in particular for real-time multimedia applications. To maintain an adequate quality of experience for end customers, we need to monitor and eventually control the effects of the network behavior. Thus, an accurate network performance model is needed, which is not a simple task, given the complex dynamics of a network. In this paper, we propose to represent the effect of an IP network on a video connection as a single G/M/1 queue. When a video session is set up, the two ends can infer the queue parameters based only on local measurements of the stream itself and with a single exchange of information between the two ends. We found that the recommended queue parameters may be different from one connection to the next but that the actual shape of the arrival process does not have a large impact on the selected parameters. In fact, using a lognormal distribution may work for many cases. Finally, we found that jitter can be used as a proxy to estimate network delay without the need for end-to-end synchronization.

## 6.1 Introduction

Multimedia services over IP (Internet Protocol), particularly video applications, have been rapidly increasing. Cisco (Cisco) predicts that by 2019, video users will generate around 80 to 90 percent of the total traffic. Content providers and network operators are increasingly competing to offer high-quality digital streamed and real-time video services both to wired and wireless users. A wide range of video streaming applications, such as video on demand (VoD) or peer-to-peer video streams (P2P), are carried by the Transmission Control Protocol (TCP). On the other hand, real-time video services, such as video conferencing, tele-medicine and live streaming often use the User Datagram Protocol (UDP) or Real-Time Protocol (RTP) over UDP.

Delivering this kind of applications through IP networks raises some challenging issues. A good quality of experience (QoE) for the user can be achieved only through strict quality of

service (QoS) requirements in terms of bandwidth, end-to-end transit delay, end-to-end jitter and packet loss. Video traffic is particularly sensitive to end-to-end jitter, that can cause some undesirable effects such as pixelation and frozen images. A number of techniques such as video compression and playback buffering mechanisms are concurrently used to manage the QoS and offer a suitable video quality.

Simple QoS management is based on measuring and collecting some performance metrics but this falls short of providing simple models to understand and recreate the behavior of the network. In our view, in order to be more effective in controlling QoS, simple and accurate modelling of the network behavior is needed. In this paper, we propose that the effect of the whole IP network on a video connection be represented by a single G/M/1 queue whose parameters are estimated through very simple measures at the two ends and a single exchange of information.

### 6.1.1 Related Work

The traditional approach for network planning and control is to model the network as a queuing system and derive from this the appropriate QoS metrics from which one can extract the relevant QoE values. It is possible to get a full characterization of network performance if the network is modelled as a network of M/M/1 or M/M/1/K queues (Kobayashi and Konheim, 1977; Rolland et al., 1999; Dahmouni et al., 2012a,b). One can get simple solutions because of the simplicity of the Poisson model. It provides a simple model for end-to-end delay, packet loss and jitter estimation, which can be used in network planning. Authors in (Mandjes et al., 1999) conclude that there are cases where it is possible to estimate end-to-end delay in core networks by modelling it with a series of M/D/1 queues.

However, there is a large body of work (Paxson and Floyd, 1995; Garcia et al., 2007) showing that multimedia traffic, and in particular video traffic, shows a long-range dependence so that the Poisson model is definitely not an accurate description of link traffic. Extending the classical techniques to non-Poisson traffic is much more difficult.

An estimation technique for the end-to-end delay is described in (Ming and Schormans, 2005). This is based on measuring the queue length at the nodes on a path. The authors assume that the input process is Gaussian, which includes a large class of self-similar process. They fit the measured distribution with the asymptotic queue length and reconstruct from this the sojourn time in the queue. The end-to-end delay is estimated by convolution.

The work of (Magri et al., 2015) gives formulas for the jitter in case of On-OFF traffic like VoIP applications with constant service time. This is based on the assumption of independent

and exponential transit times for consecutive packets. This is extended in (Huremovic and Hadzialic, 2015) to the case of a network path using the techniques first proposed in (Dahmouni et al., 2012a).

Instead of trying to derive the QoS parameters of a session from the network links, a different approach is to monitor the session itself and to infer from this the relevant QoS parameters. Most of these techniques are based on the assumption that the behavior of the session is determined by a single bottleneck link where congestion occurs. Many techniques, such as the packet-pair method, also inject artificial traffic on the session to estimate its performance. See (Hu and Steenkiste, 2003) for a discussion of these techniques.

Other authors, as in (Coates and Nowak, 2000; Alouf et al., 2001), try to derive an equivalent queue model based on a single bottleneck in a M/M/1/K or M/D/1/K queue. They use standard queuing theory to extract some performance metrics. For the M/M/1/K queue, they can get a large number of metrics such as packet loss, bandwidth capacity and the background traffic intensity on the link. This is much more difficult for the M/D/1/K where only a few metrics can be derived, and with considerable numerical effort. These are then compared to actual end-to-end network measurements to determine the accuracy of the model. They conclude that M/M/1/K queue gives a reasonable results for estimating these metrics.

### 6.1.2 Contribution

The main contribution of this paper is to propose an equivalent G/M/1 queue to model network performance that can estimate fairly accurately the network transit time and jitter using only very simple local measurements. We use jitter as a proxy to estimate those measurements without the need for end-to-end synchronization. This is based on some previous results on network jitter evaluation, combined with the analysis of traffic traces for TCP and UDP traffic. Differently from (Coates and Nowak, 2000) and (Alouf et al., 2001), we don't assume a bottleneck link and don't use artificial traffic either. The constraint is that only measurements that are locally available can be used so that the clocks at the two ends need not be synchronized. The tradeoff is between the amount of information measured and transferred between the users on the one hand and the assumptions that are needed to arrive at a queuing model on the other hand.

There are some advantages in having a credible queuing model for the network. First, we can compute the potential impact that a change in the source parameters could have on some performance metrics such as delay and jitter. Conversely, a queuing model can help us decide whether an observed change in some performance measure is simply due to the stochastic

nature of the queue or whether network conditions have changed. In that case, it is possible to express this as a change in the parameters of the queue and use this to control the source or the receiver rates.

### 6.1.3 Paper Structure

This process can be summarized as follows. First, we propose in section 6.2 a G/M/1/ queue with infinite buffer and First-Come First-Served (FCFS) service and discuss some assumptions related to this queue.

Next, we consider in section 6.3 streaming video over TCP. We propose an algorithm where the two ends of the session can make measurements of the session traffic and from this, infer a *"best"* equivalent queue.

We then briefly discuss in section 6.5 how this algorithm could be used for real-time traffic carried over UDP. We mention that the algorithm cannot be used directly with raw UDP but that using RTP would provide enough information to identify the equivalent queue.

Finally, we discuss in section 6.6 the relatively small impact of the shape of the distribution of the inter-arrival time on jitter and show that it depends mostly on first two moments of the distribution.

## 6.2 The G/M/1 Queue

In order to identify a queue, we need four elements :

1. The service discipline. Here, we assume FIFO
2. The buffer size.

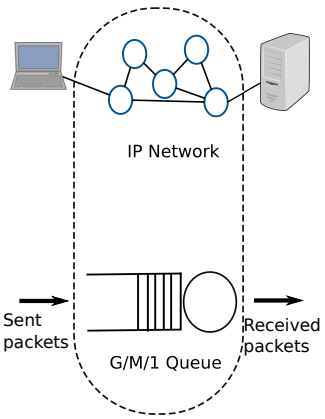



Figure 6.1 Network G/M/1 Queue Equivalent

3. The arrival process
4. The service process

First, we assume that the queue equivalent to the whole network path, both core and access, is a G/M/1 queue, as shown in figure 6.1. This model has two important features. One is that given the arrival process $G$ and the sojourn time distribution, we can compute the service rate exactly. The other is that given this service rate, one can compute the mean jitter analytically. We think that this assumption is acceptable for the following reasons.

### 6.2.1 Infinite Buffer

We assume that we have no losses so that we can assume that the buffer is infinite. This assumption is based on the measurements made on some real video sessions where we find that the packet loss is in fact very small and can be neglected. Note that for a real-time service, a packet that is delayed more than a certain time, either because of network congestion or retransmission, is considered lost since it cannot be used to reconstruct the video frame once this frame has been sent to the application.

### 6.2.2 Generic Arrival Process

There is clearly an advantage in modeling the traffic entering the network as an arbitrary process with a general distribution for the packet inter-arrival time. This avoids the Poisson assumption which is not considered realistic for multimedia traffic (Paxson and Floyd, 1995; Crovella and Bestavros, 1997; Garcia et al., 2007), at least in the core.

The discussion so far has assumed that we know the complete inter-arrival time distribution $f_R$ at the source. In practice, this may not be known and we may have to measure the process to get an estimate of the distribution. This is complicated and may take some time to get reliable statistics. To simplify the procedure, we make a further assumption that *The source inter-arrival time can be modelled by a two-parameter distribution.* This is a relatively strong assumption but as we will see, it turns out to be reasonably accurate for the sources we have considered here.

### 6.2.3 Exponential Service Time

A much stronger assumption is that of an exponential service time. We think that this is not unrealistic for the following reasons.

First note that the service time in question is *not* the service time at a particular network queue. Rather, it is an aggregated measure of the time spent in the network : It is the sum

of all the waiting and service times experienced by a packet as it crosses the network. The reason why a Poisson model may be accurate is that on each link, the waiting time of a packet depends on all the cross-traffic that share the queue of the packet under consideration. Two consecutive packets of the same stream may be separated by a number of packets from other streams. To the extent that these background streams are all different and uncorrelated, it is not unreasonable to assume that the total time in the queue for one packet is in effect driven by a Poisson process.

Another reason why the exponential assumption may be realistic is based on the work of (Alouf et al., 2001) where an the exponential service time seems to be quite a good approximation for end-to-end service time through the network.

Finally, we have made some measurements on some real streaming video flows carried over TCP. We have measured the Round-Trip Time (RTT) using the ACK packets from the client to the server and plotted the corresponding histogram with a fitted exponential. Some of our results are shown on figures 6.2 and 6.3. We can see that the agreement is quite good except at low values of the transit delay where the exponential distribution is clearly over-estimating the frequency of short delays. Based on this and other similar results not presented here to conserve space, we use the Poisson service time model in the following.

## 6.3 Estimation Algorithm for Streaming over TCP

The question that we examine in this section is how the two ends of the connection can use this information to arrive at a consistent estimation of the queue parameters.

At first, we examine a relatively frequent case where video is streamed over a TCP connection. The traffic from the source to the client is made up mostly of data packets while the traffic

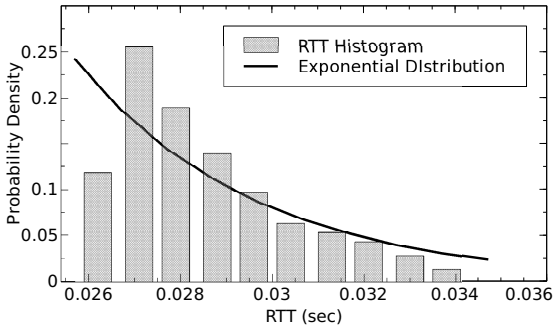



Figure 6.2 Approximate Packet Transit Time Vs Exponential Distribution- V-2 Experiments

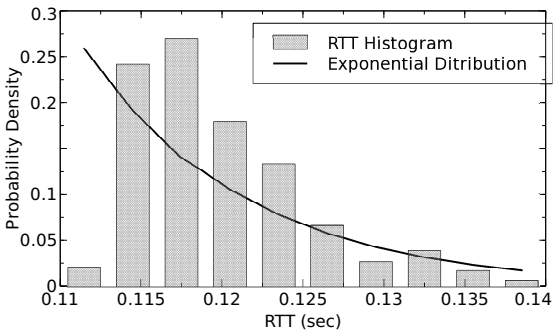



Figure 6.3 Approximate Packet Transit Time Vs Exponential Distribution- V-5 Experiments

in the other direction is made up mostly of acknowledgement (ACK) packets from the client to the server.

### 6.3.1 Estimation Procedure

We present the estimation algorithm where the computation is made at the source but it could be obviously done at the receiver as well. In all that follows, we try to keep the measurements as few and as simple as possible and to limit to the maximum the amount of information that has to be transmitted between the source and the destination.

First, we define variables that are used by this algorithm

$J_m$ Measured Jitter

$t_i$ time at which the packet $i$ is sent by the source

$r_i$ time at which the packet $i$ is received at the destination

$R_i$ inter-arrival time, $R_i = t_i - t_{i-1}$

$O_i$ inter-departure time, $O_i = r_i - r_{i-1}$

$f_R$ probability density function (pdf) of inter-arrival time variable $R$

$f_R^r$, log-normal pdf, gamma pdf, Pareto Type I pdf

$r \in \{l, g, p\}$

$\hat{J}^r$, analytic jitter corresponding ti pdf $r$

$r \in \{l, g, p\}$

The client measures the average jitter of the arriving packets. We get the $i$th packet's sending and receiving instants $t_i$ and $r_i$ and we take the mean over all $N$ packets

$$J_m = \frac{1}{N} \sum_{j=1}^{N} |(t_i - t_{i-1}) - (r_i - r_{i-1})|$$

$$= \frac{1}{N} \sum_{j=1}^{N} |R_i - O_i|. \tag{6.1}$$

This is possible because of the definition of jitter (6.11) given in Appendix 6.9.3. In this equation, the values of $r_i$ is known from the local clock and that of $t_i$ from the TCP time stamp. The jitter is simply the difference between the inter-arrival time at the source minus that at the destination so that the synchronization offset between the two ends simply cancels out.

The source, on the other hand,

1. Measures the mean and standard deviation of the inter-arrival process
2. Measures the average transit time
3. Assumes some two-parameter distribution for the inter-arrival time distribution selected from some pre-defined list. For each distribution in the list
   (a) Computes the two parameters of the distribution, generally the location and scale
   (b) Computes the average service time for the queue
   (c) Computes the jitter for the corresponding queue
   (d) Compares the computed jitter with the measured value from the client
4. Chooses the distribution that has the smallest difference between the computed and the measured value of jitter.

We discuss each step and show how it can be done and the amount of work that is needed in each case.

**Measuring the Arrival Parameters**

Step 1 is to compute the mean $m$ and standard deviation $\sigma$ of the packet inter-arrival time using information on the departure times $t_i$ from the server. We have

$$m = \mathrm{E}(R_i) = \mathrm{E}(t_{i+1} - t_i) \tag{6.2}$$

$$\sigma = \sqrt{\mathrm{var}(R_i)} \tag{6.3}$$

**Measuring the Transit Delay**

In principle, measuring the packet transit delay Step 2 is straightforward : Simply put a time stamp at the source and compare with the arrival time at the destination. This of course

assumes that the two clocks are synchronized to a sufficient accuracy. The only practical way of doing this is using the Network Time protocol (NTP) but the accuracy of these measurements can range from a few tens of milliseconds over a network connection up to 100 milliseconds or more on asymmetric routes and in the presence of network congestion. This is not good enough when dealing with network delays of a few tens of milliseconds. For this reason, we use only the RTT where the server uses the `Tsecr` field of the ACK packets which contains the time when the packet being acknowledged left the server. The RTT is then the difference between the time when ACK packet is received back at the server and the departure time of that acquitted packet. From this, one can easily compute the average transit time.

### Selecting a Candidate Inter-arrival Distribution

The step 3 is to choose a two-parameter distribution $f_R$ for the inter-arrival time from some given small set. In this paper, we have used three distributions, log-normal, gamma and Pareto Type I. The corresponding pdf are denoted $f_R^l$, $f_R^g$ and $f_R^p$.

### Compute the Distribution Parameters

Given the values of $m$ and $\sigma$, the server can compute in step 3a the scale and location parameters of $f_R$. In some cases, such as the gamma distribution, this can be done in closed form. If not, a numerical procedure is needed to solve a nonlinear system of two equations in two variables.

### Computing the Average Service Rate

With these informations, the server can compute in step 3b the average service time. For this, we use the fact that the transit time and the service times are related to the Laplace transform of $f_R$ given by (6.8) in appedix 6.9.2. It is generally not possible to do this analytically and we must use a numerical solver for a set of two nonlinear equations in two variables to get the value of $\mu$. At this point, we have all the parameters of the equivalent queue corresponding to the $f_R$ that was chosen.

### Computing Jitter

Once we have the equivalent queue, we can then use the results of (Dbira et al., 2016a) to compute the average jitter $\hat{J}$ for this queue given by (6.12) in appendix 6.9.3. In some cases, we can compute an analytic expression for the jitter. If not, this can be done by a numerical

integration technique. We can write $\hat{J}^r$, $r \in \{l, g, p\}$ to indicate that the value of $\hat{J}$ depends on the choice of $f_R$.

**Selecting a Model**

The server repeats steps 3a–3d for a number of arrival distribution with corresponding jitter values $\hat{J}^l$, $\hat{J}^g$ and $\hat{j}^p$. Details about the analytical formula are presented in appendix 6.9.4. These values are then compared with the actual average jitter value $J_m$ measured by the client and transmitted to the server. The distribution that has the best smallest absolute difference between the computed and measured jitter defines the equivalent queue.

### 6.3.2 Summary

The procedure outlined above is such that for a given TCP session in a stationary state and with low loss, the two ends will be able to identify an equivalent queue that 1) has the same first two moments as the arrival distribution, 2) has the same average transit delay and 3) has the the best fit for the jitter measured at the destination among the set of potential arrival processes. The algorithm methodology is summarized in figure 6.4.

## 6.4 Comparing Video Streams

The technique we propose here will provide a best guess, in the sense defined above, for the service time and inter-arrival time distributions of the queue. In this section, we compare the equivalent queues obtained for different experimental streams.

### 6.4.1 Experimental Setup

We have used two different configurations. The simplest is where the server and the client are directly connected to a home router, as shown on figure 6.5. While this configuration does not represent in any way a network connection, it is shown here as a benchmark to measure the impact of the access network on the measurements. The other configuration is through the Internet as shown on the top of figure 6.1.

For each experiment, we use about 10 minutes of two MPEG-4 videos which produce traces of 50 000 packets. The streams contain a mixture of scenes, some with more changes than other. These are streamed using the `Videolan` application both as the server and the client and are transported over TCP.

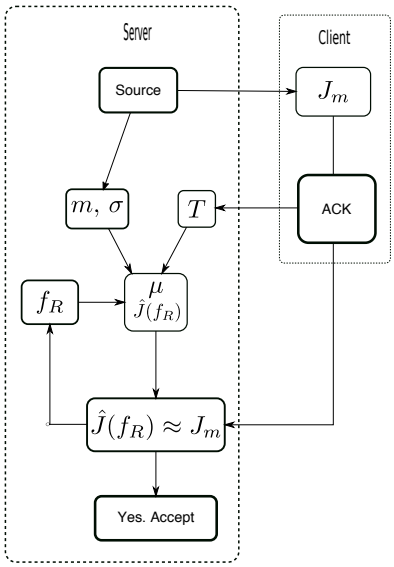



Figure 6.4 Verification Procedure

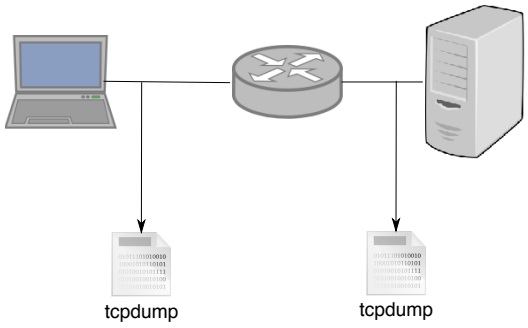



Figure 6.5 Video Transmission Test-bed in LAN and WAN

In practice, the algorithm that we described would need the source and destination applications to make the various measurements and calculations required by the model. Because we did not want to modify `Videolan`, we captured the packet traces using `TCPdump` on the wifi or ethernet port of the source and the destination computers. Both traces are analyzed off-line using standard statistical tools to get the values needed by the model.

Except for the first experiment on a single router, the client and server were located in different places. Many were in the Montreal area but we were able to get some measurements from some widely separated locations. The details are given in table 6.1.

### 6.4.2 Stream Measurements

For each stream, we measure four quantities : the mean and the standard deviation of the arrival process, the mean transit time $T = \frac{RTT}{2}$ and the measured jitter $J_m$. These are summarized in table 6.2 and they are given in milliseconds. One can see that for a given video source, the measured values of the mean $m$ are reasonably close to each other but that those for the standard deviation $\sigma$ show much larger variations due to the other processes running on the machine. The differences in the transit times and jitter reflect the conditions of the network at the time the measurements were made.

### 6.4.3 Model Selection

For each measurement of table 6.2, the procedure will yield a best queue based on the difference between the calculated jitter for each distribution $\hat{J}^r$ $r \in \{l, g, p\}$ and the actual measurement $J_m$. This jitter values are compared in table 6.3. The shaded cells of the table show the distribution selected by the algorithm. The interesting point is that in most cases, the log-normal or the gamma distributions are selected. In fact, the log-normal fits many cases and there is a small difference in the other cases.

Table 6.1 Video Streaming over TCP Experimental Tests Summary

| Experiment | Client Access Network | Server Location | Client Location |
|---|---|---|---|
| V-1 | WiFi | Montreal | Montreal |
| V-2 | WiFi | Waterloo | Montreal |
| V-3 | WiFi | Montreal | Montreal |
| V-4 | $3G^+$ | Montreal | Tunis |
| V-5 | WiFi | Milan | Montreal |

Table 6.2 Measurements of Video Stream Parameters in milliseconds (msec)

| | Video 1 | | | | Video 2 | | | |
|---|---|---|---|---|---|---|---|---|
| | $m$ | $\sigma$ | $T$ | $J_m$ | $m$ | $\sigma$ | $T$ | $J_m$ |
| V-1 | 6.16 | 43.34 | 1.985 | 0.52 | 13.80 | 45.00 | 2.35 | 4.07 |
| V-2 | 6.13 | 38.48 | 10.02 | 2.015 | - | - | — | — |
| V-3 | 6.99 | 33.35 | 16.23 | 2.78 | 9.18 | 39.60 | 15.86 | 4.15 |
| V-4 | 7.85 | 66.15 | 103.93 | 12.95 | 7.29 | 37.14 | 86.7 | 5.07 |
| V-5 | 6.98 | 31.90 | 147.8 | 9.28 | 8.29 | 36.80 | 83.64 | 4.69 |

Table 6.3 Experimental END-TO-END Jitter and Analytical Jitter Comparison for TCP Traffic

| | Video 1 | | | | Video 2 | | | |
|---|---|---|---|---|---|---|---|---|
| Experiment | $J_m$ | $\hat{J}^l$ | $\hat{J}^g$ | $\hat{J}^p$ | $J_m$ | $\hat{J}^l$ | $\hat{J}^g$ | $\hat{J}^p$ |
| V-1 | 0.52 | 0.98 | 0.34 | 1.71 | 4.07 | 1.70 | 1.07 | 0.01 |
| V-2 | 2.015 | 2.628 | 1.38 | 3.53 | — | — | — | — |
| V-3 | 2.78 | 3.80 | 2.7 | 4.45 | 4.15 | 4.39 | 3.18 | 5.42 |
| V-4 | 12.95 | 9.78 | 5.5 | 6.33 | 5.07 | 6.73 | 7.02 | 5.91 |
| V-5 | 9.28 | 7.55 | 8.3 | 6.21 | 4.69 | 7.43 | 7.94 | 6.49 |

It will also produce a different mean service time in each distribution case $S^r$ with $r \in \{l, g, p\}$, as can be seen in table 6.4. So that the queue utilization $\rho^r$, $r \in \{l, g, p\}$ is also very different as shown in table 6.5.

The results of table 6.3 raise an interesting possibility. Instead of trying to guess the distribution for the arrival process, one might choose one distribution for *all* streaming video over TCP. This is a very strong assumption that would need to be validated over a large set of sessions, something that is clearly outside the scope of this paper. Using the results of table 6.3, we compute the overall root-mean-square error between the measured and computed values for all distributions. This value is given in table 6.6 for the three distributions, where we have

Table 6.4 Estimated Service Time (ms)

| | Video 1 | | | Video 2 | | |
|---|---|---|---|---|---|---|
| Experiment | $S^l$ | $S^g$ | $S^p$ | $S^l$ | $S^g$ | $S^p$ |
| V-1 | 0.8624 | 0.1848 | 1.8172 | 1.656 | 0.759 | 2.346 |
| V-2 | 2.2068 | 0.7969 | 4.1684 | — | — | — |
| V-3 | 3.2154 | 1.6077 | 5.2425 | 3.8556 | 2.0196 | 6.6096 |
| V-4 | 4.1605 | 3.14 | 7.536 | 6.0507 | 4.3011 | 6.6339 |
| V-5 | 5.933 | 4.9907 | 6.7008 | 6.2175 | 4.4766 | 7.5439 |

Table 6.5 Estimated Utilization

| | Video 1 | | | Video 2 | | |
|---|---|---|---|---|---|---|
| Experiment | $\rho^l$ | $\rho^g$ | $\rho^p$ | $\rho^l$ | $\rho^g$ | $\rho^p$ |
| V-1 | 0.14 | 0.03 | 0.295 | 0.12 | 0.055 | 0.17 |
| V-2 | 0.36 | 0.13 | 0.68 | — | — | — |
| V-3 | 0.46 | 0.23 | 0.75 | 0.42 | 0.22 | 0.72 |
| V-4 | 0.53 | 0.4 | 0.96 | 0.83 | 0.59 | 0.91 |
| V-5 | 0.85 | 0.715 | 0.96 | 0.75 | 0.54 | 0.91 |

excluded the first row since it does not correspond to a real network connection. We can see that the agreement is better for the log-normal distribution than for a gamma or Pareto so that would be a good candidate.

Note however that while the log-normal seems more accurate, using the gamma would be faster. This is because we can express the parameters $k$ and $\theta$ analytically in terms of the mean and standard deviation and also because we have a closed form expression (6.23) for the jitter $\hat{J}^g$ (appendix 6.9.4).

## 6.5 Real-Time Video over UDP

The case of real-time video is different since these connections often use UDP instead of TCP and the traffic flow is similar in both directions. Instead of having one node identified as the sender and the other as the receiver, as with TCP, in the present case, both end points can be viewed as a sender for its outgoing traffic and a receiver for the incoming packets. In this section, we verify to what extent the G/M/1 model carries over to video over UDP.

### 6.5.1 Computing the Queue Parameters

Consider a flow from the one node, called the sender, to the other, called the receiver. The algorithm of section 6.3.1 needs some measurements, some of which are not possible with UDP. Step 1 of the algorithm needs an estimate of the mean $m$ and standard deviation $\sigma$ of the arrival process, which can be measured by the source. Step 2 needs an estimate of the average transit time. For UDP, this information is not available from the packets since they

Table 6.6 Root-mean square error

| Log-normal | Gamma | Pareto |
|---|---|---|
| 2.48 | 4.9 | 4.6 |

carry no timing information. One possible estimation is to use the first sent-received packet pair exchange to get this estimate (Rossi et al., 2009). The calculations in steps 3a–3c can also be carried out as with TCP.

The main problem is the fact that the receiver cannot compute the actual jitter in (6.1) since the UDP packets contain no timing information or sequence numbers. This means that the algorithm cannot be used for UDP under the assumption we made about the available information.

Still, we think it is worth checking if the technique proposed in section 6.3 used for TCP would give good results for UDP it ever were possible to implement it with some future expanded version of UDP or if the required information could be carried on RTCP packets from the RTP protocol.

### 6.5.2 Experimental Setup

In order to do this, we use the same LAN/WAN expriment configuration as shown in figure 6.5. An important real-time video service carried over UDP is video-conferencing so that we chose to use `Skype`, which is one of the most popular video-conferencing platform. Here again, the `Skype` application was installed on two machines. Details about the experiments are presented in table 6.7. The test *S-1* was on a residential local network while for *S-2* and *S-3*, the machines were located on distant networks as shown in table 6.7. We ran the video-conference for around 10 minutes so that we capture more than 50 000 UDP packets per experiment.

We must assume also that the receiver has a way to identify packets at both ends and the time where they were sent or received. We can do this by using the `tcpdump` trace at the sender and the receiver since these traces contain a time stamp for each packet. We can uniquely identify the packet from the data field so that it is possible for the receiver to compute the actual jitter, something that is not possible based only on the actual UDP packet contents.

Table 6.7 Video-Conferencing over UDP Experimental Tests Summary

| Experiment | Client Access Network | Client 1 Location | Client 2 Location |
|---|---|---|---|
| S-1 | WiFi | Montreal | Montreal |
| S-2 | WiFi | Montreal | Montreal |
| S-3 | 3G | Tunis | Montreal |

### 6.5.3 Estimation Algorithm

The parameters of the video source are presented for the `Skype` traffic in table 6.8. In the present case, the sources are different since they depend on the actual contents of the conversation and the changes in the image. The main difference seems to be the much smaller standard deviation when compared with the video streams. As one could expect, the mean transit time for S-1 is much smaller than for transmission over a real network since the client and the server are on the same LAN. Overall, these parameters are comparable to those for TCP even though the sources are quite different.

The final step is to compare the measured $J_m$ with the analytical values computed from different distributions given by $\hat{J}^l$, $\hat{J}^g$ and $\hat{J}^p$. These are presented in table 6.9 where we have indicated the best fit in gray. We can see that here also the best fit is mostly the log-normal or gamma distribution and then with a relatively small difference between the two. We also present in table 6.10 the values for the queue utilization. These also cover a wide range from very low to almost saturated.

## 6.6 Impact of the Arrival Process

The results of the previous sections indicate that the G/M/1 queue presents a good model for an end-to-end video connection either for TCP or for UDP protocols. We have seen the that the measured jitter $J_m$ is quite close to most of the values of the analytical jitter $\hat{J}(f_R)$. This seems to suggest that the jitter in a G/M/1 queue depends mostly on the mean $m$ and standard deviation $\sigma$ of the inter-arrival process $R$ and the shape of the distribution does not matter all that much.

We examine this idea that the jitter does not depend strongly on the particular shape of the inter-arrival distribution by simulation. We measure the jitter in a G/M/1 queue with a different type of distribution such as Pareto ($Pr$), log-normal ($LogN$), gamma ($Gm$) and tri-modal ($TriM$). The tri-modal distribution combines 10% of pareto random variables, 60% log-normal and the other 40% are gamma. For each simulation, we fix the standard deviation $\sigma$ of $R$ to the values 0.5, 2 and 4. For each case, we plot in figures 6.6, 6.7 and 6.8 the jitter

Table 6.8 Source Measurements (ms)

| Experiment | $m$ | $\sigma$ | $T$ |
|---|---|---|---|
| S-1 | 14.36 | 41.62 | 4.03 |
| S-2 | 3.52 | 13.9 | 30.7 |
| S-3 | 5.73 | 7.66 | 42.5 |

Table 6.9 Experimental and Analytical Jitter (ms)

| Experiment | $J_m$ | $\hat{J}^l$ | $\hat{J}^g$ | $\hat{J}^p$ |
|---|---|---|---|---|
| S-1 | 3.15 | 2.29 | 1.04 | 2.01 |
| S-2 | 2.10 | 2.54 | 2.37 | 2.49 |
| S-3 | 6.16 | 4.38 | 4.9 | 4.05 |

Table 6.10 Utilization

| Experiment | $\rho^l$ | $\rho^g$ | $\rho^p$ |
|---|---|---|---|
| S-1 | 0.11 | 0.28 | 0.14 |
| S-2 | 0.58 | 0.4 | 0.82 |
| S-3 | 0.79 | 0.75 | 0.87 |

as a function of the traffic load $\rho = \lambda/\mu$ where the service time $1/\mu$ is chosen as the time unit. We can see from these results that for a small standard deviation, the shape of the distribution has very little effect on the jitter and this for the whole range of traffic loads.

Results are somewhat different for the two other cases, as can be seen in figures 6.7 and 6.8. The impact of the distribution shape is still small for small traffic load but increases somewhat with the load. Still, we can see that the actual differences are not all that large, even close to saturation.

This result suggests that for the G/M/1 queue, only the inter-arrival mean and standard deviation might be required to identify the queue. This reduces the queue selection process to the estimation of the moments of $R$ for some chosen distribution and the calculation of the jitter for this distribution.

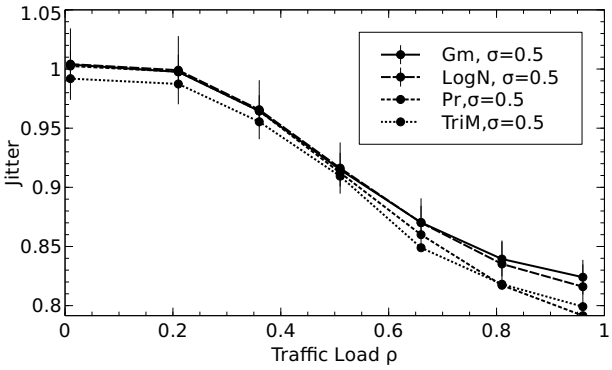



Figure 6.6 Jitter in a G/M/1 Queue for $\sigma = 0.5$

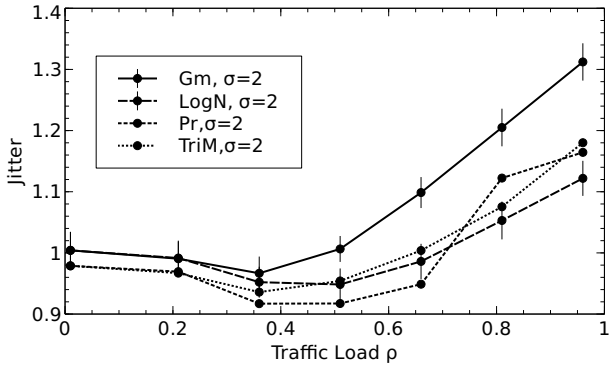



Figure 6.7 Jitter in a G/M/1 Queue for $\sigma = 2.0$

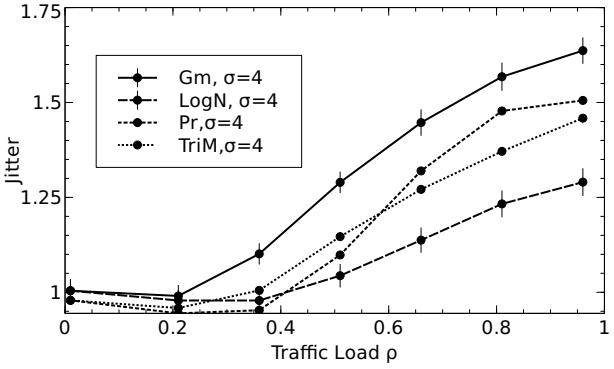



Figure 6.8 Jitter in a G/M/1 Queue for $\sigma = 4.0$

## 6.7 Conclusion

A new queue model to study the behavior of IP network video applications was proposed. We found that a single G/M/1 queue can be used as an equivalent queue to study end-to-end network performance of a video session. Moreover, it was found that the log-normal probability density function for packet inter-arrival time gives the best fitting results for end-to-end performance, in particular for jitter. A contribution of this work was precisely to use jitter as a proxy to estimate network delay. This is particularly interesting because, contrary to other latency measurements, jitter measures does not require end-to-end synchronization.

After choosing jitter as a verification tool, we proceeded to compare network jitter in experimental and simulation environments with the results of a G/M/1 queue. We showed that the use of the G/M/1 queue is valid and that, in many cases, the jitter in a queue depends strongly on the mean and standard deviation of the inter-arrival time distribution but not so much on the actual shape of the distribution.

These results are very important and useful for managing QoS for video applications. They yield a good model for over-the-top services that only depend on the video provider and video client, regardless of the details of the network infrastructure. It also gives an analytical platform to determine total service time through the network. Finally, it provides a simple and analytic expression to estimate network jitter for a non-Poisson traffic, which is highly useful for QoS-oriented optimization.

## 6.8 Acknowledgements

This work was supported by Grant No. 121949-2012 from the Natural Sciences and Engineering Research Council of Canada.

## 6.9 Appendix

### 6.9.1 Definition of Variables

Based on the above assumption, we now define the variables that we will be using. We define for the $i^{th}$ packet of a particular video stream

$\lambda$ arrival rate, $\lambda = 1/E[R]$

$S_i$ service time,

$\mu$ service rate, $\mu = 1/E[S]$

$W_i$ waiting time in the queue before service

$T_i$ transit time through the queue, $T_i = W_i + S_i = r_i - t_i$

$\eta$ transit rate $= 1/E[T]$

We denote $f_R$, $f_S$ and $f_T$ are respectively the probability density functions (PDF) of $R$, $S$ and $T$.

### 6.9.2 Properties of the G/M/1 Queue

In order to characterize the queue, we need a description of the arrival and service processes. The arrival process is defined by the inter-arrival time distribution $f_R$ of packets at the source. We assume that this is known, either because the codec output process is known, or by direct measurement by the source.

The service process is exponential by definition so that the only remaining unknown is the service rate $\mu$ that must be estimated from some measurements. For this, we make use of a particular feature of the G/M/1 queue where the service rate $\mu$ is directly related to the transit time $\eta$ through the queue.

First, recall (Kleinrock, 1975) that the transit time $T$ in a G/M/1 queue has an exponential distribution with parameter $\eta$ given by

$$f_T(x) = \eta e^{-\eta x} \tag{6.4}$$

$$\eta = \mu(1 - \tau) \tag{6.5}$$

where $\tau$ is the probability that a packet will have to wait before entering the server and it is given by the root of

$$\tau = \mathcal{F}_R(\mu - \mu\tau) \tag{6.6}$$

where

$$\mathcal{F}_R(s) = \int_0^\infty f_R(y) e^{-sy} dy \tag{6.7}$$

is the Laplace transform of $f_R(y)$.

We see that $\eta$ and $\mu$ are directly related by the following nonlinear system (6.5) and (6.6)

$$\begin{cases} \eta = \mu(1 - \tau) \\ \tau = \mathcal{F}_R(\mu - \mu\tau). \end{cases} \tag{6.8}$$

### 6.9.3 Jitter Definition

The variation of delay in a tagged sequence of packets over time goes under different terms like *jitter*, *end-to-end jitter*, or simply *delay variation*. In the following, we will be using the term *jitter* to mean any of these.

Standards organizations give different formal definitions for jitter. The International Telecommunication Union-Telecommunication Standardization Sector (ITU-T) defines jitter as the variation of delay from some reference value such as the mean or minimum delay (itu, 2011; Clark, 2003). It is the difference between the end-to-end delay, i.e., *latency* of a packet $i$, $T_i$, and some reference delay, $a_i$ for a given packet flow

$$J = E\left[|T_i - a_i|\right]. \tag{6.9}$$

With this definition, it is possible to quickly detect any increase in packet delay.

The Internet Engineering Task Force (IETF) defines the jitter as the difference between two measurement points (Demichelis and Chimento, 2002; Angrisani et al., 2013) for a specific packet flow by $G = T_i - T_{i-1}$. The end-to-end jitter is defined as the expected absolute value of random variable $G$.

$$J = E\left[|T_i - T_{i+1}|\right]. \tag{6.10}$$

The jitter can be measured without synchronization if we write

$$\begin{aligned} G_i &= T_i - T_{i-1} \\ &= r_i - t_i - (r_{i-1} - t_{i-1}) \\ &= (r_i - r_{i-1}) - (t_i - t_{i-1}) \\ &= O_i - R_i \end{aligned} \tag{6.11}$$

In other words, the jitter is simply the difference between the inter-arrival time at the destination and at the source for a given packet. We can measure these locally at each end of the connection without having to synchronize them. We can then compute the distribution of $G$ from the difference if we can reliably identify packets in the two ends.

The second requirement is that we should be able to compute the jitter after identifying some queue parameters. Recent work (Dahmouni et al., 2012a; Dbira et al., 2016a) has provided accurate and fast computation models for estimating $J$ derived from equation 6.10.

The fact that for a G/M/1 queue both service and transit times are exponential allows us to derive an exact expression for the jitter that depends only on $\mathcal{F}_R$ (Dbira et al., 2016a)

$$\hat{J} = \frac{(\eta^2 + \mu^2)}{\eta\mu(\eta + \mu)} + \frac{2}{(\eta + \mu)}\mathcal{F}_R(\eta + \mu) - \frac{1}{\eta}\mathcal{F}_R(\eta). \tag{6.12}$$

Authors in (Dbira et al., 2016a) present simplified expressions for jitter for different inter-arrival time distributions like for Deterministic, Gamma and Pareto distributions.

### 6.9.4 Properties of Some Distributions

Here we review the definition of the pdf and the expression of the mean and standard deviation in terms of the parameters for the distributions we have used for the inter-arrival process. We also give some closed form expressions for the jitter when these are available.

#### Log-normal

$$f_R^l(y; \theta, \omega) = \frac{1}{y\sqrt{2\pi}\theta} \exp\left(-\frac{(\ln y - \omega)^2}{2\theta^2}\right) \tag{6.13}$$

$$m = e^{\omega + \theta^2/2} \tag{6.14}$$

$$\sigma = \sqrt{(e^{\theta^2} - 1)e^{2\omega + \theta^2}} \tag{6.15}$$

In this case, we cannot get an expression for the Laplace transform and we must use numerical integration

$$\mathcal{F}_R^l(s) = \int_0^\infty e^{-sy} f_R^l(y; \theta, \omega) dy \tag{6.16}$$

The value $\mathcal{F}_R^1(\eta)$ can then be used in (6.12) to get the jitter.

#### Gamma

This is the simplest case from a computational point of view.

$$f_R^g(y; k, \theta) = y^{k-1}\frac{e^{-y/\theta}}{\theta^k\Gamma(k)} \tag{6.17}$$

$$m = k\theta \tag{6.18}$$

$$\sigma = \sqrt{k\theta^2} \tag{6.19}$$

From this, we can get a closed form solution for the parameters $k$ and $\theta$ in terms of the mean and standard deviation

$$\theta = \frac{\sigma^2}{m} \tag{6.20}$$

$$k = \left(\frac{m}{\sigma}\right)^2 \tag{6.21}$$

The Laplace transform can be evaluated analytically

$$\mathcal{F}_R^g(s) = \frac{1}{(1+s\theta)^k}. \tag{6.22}$$

Next, the simplified analytic expression for jitter is

$$\hat{J}^g = \frac{(\eta^2+\mu^2)}{\eta\mu(\eta+\mu)} + \frac{2}{(\eta+\mu)}\frac{1}{(1+(\eta+\mu)\theta)^k} - \frac{1}{\eta}\frac{1}{(1+\eta\theta)^k}. \tag{6.23}$$

**Pareto**

$$f_R^p(y;\alpha,m) = \begin{cases} \alpha\dfrac{x_m^\alpha}{y^{\alpha+1}} & \text{if } y \geq x_m \\ 0 & \text{otherwise.} \end{cases} \tag{6.24}$$

$$m = \begin{cases} x_m\dfrac{\alpha}{\alpha-1} & \text{if } \alpha > 1 \\ \infty & \text{if } \alpha \leq 1 \end{cases} \tag{6.25}$$

$$\sigma = \begin{cases} \frac{x_m}{(\alpha)}\sqrt{\dfrac{\alpha}{(\alpha-2)}} & \text{if } \alpha > 2 \\ \infty & \text{if } \alpha \leq 2 \end{cases} \tag{6.26}$$

The Laplace transform is given by

$$\mathcal{F}_R^p(s) = \alpha E_{\alpha+1}(sx_m) \tag{6.27}$$

which we can replace in (6.12) to get the jitter. Note that, $E_n(x)$ is the exponential integral

$$E_n(x) = \int_1^\infty \frac{e^{-xt}}{t^n}dt. \tag{6.28}$$

and from this it is possible to compute numerically $\hat{J}^p$.

# CHAPITRE 7 ARTICLE 4 : A SERVER-SIDE ADAPTIVE CONTROL FOR VIDEO STREAMING

Recopié avec permission : H. Dbira, A. Girard et B. Sansò, *A Server-Side Adaptive Control for Video Streaming*, soumis aux IEEE Transactions on Multimedia.

**Abstract**

Adaptive video streaming over HTTP is being increasingly used to enhance both users' experience and resource utilization. This is made possible by switching the quality levels of the video dynamically based on signals received from the client triggered by changes in the buffer occupancy.

In a streaming HTTP session, the traffic from the client to the server is made up almost exclusively of ACK packets which contain a measure of the round-trip time of the packet. We want to evaluate the feasibility of implementing a level selection algorithm in the video server based on that information.

Accordingly, we provide the server-side methodology for the analytical estimation of QoE impairment, in particular jitter buffer performance. Then, an online level switching decision is proposed based on this estimation model. We use simulation to evaluate the performance of the proposed method. We further compare this results with *Microsoft IIS Smooth Streaming (MSS)* algorithm which is a client buffer-based algorithm.

The performance criterion is the ability to keep buffer underflow, which is the most important QoE parameter, under a desired threshold. We found that the proposed algorithm reduces the buffer underflow probability and offers a better quality compared to MSS.

## 7.1 Introduction

Recent years have seen the dominance of video traffic over Internet. It accounts for 80% of the global Internet traffic in 2016 (Cisco) and about 40% of mobile traffic. Additionally, with the evolution of encoding techniques, video over IP (Internet Protocol) is now provided in High Definition and Ultra High Definition to satisfy users' demand.

Current video-streaming applications are increasingly using HTTP/TCP, due to the emergence of HTTP-based Content Distribution Networks (CDN). Billions of videos offered by various content providers like *Netflix, Akamai and Youtube* are streamed every day over

HTTP (Stockhammer, 2011). The main characteristics of video streaming over HTTP are reliability, simplicity, high availability and adaptability with mobile networks (Stockhammer, 2011). However, it is noteworthy that TCP is not well suited to carry video traffic with time constraints. This is because TCP uses retransmission mechanisms, which can affect packet latency and jitter and damage the video quality. Therefore, the challenging task is to maximize the quality of the video streaming service given the network condition. To match the stochastic nature of the network, the streaming is becoming adaptive.

Current adaptive video streaming algorithms are mostly client-based. The server always waits for reports about the quality of the connection before selecting the quality of the next video segment from the player. However, it is known that TCP acknowledgements (Ack) carry some information about network QoS such as the round-trip time (RTT) (Jacobson et al., 1992). This point is already used by (Dbira et al., 2017) to carry out a server-side network jitter estimation. Note that in a streaming connection, the flow from the server to the client is made up of data packets while the reverse flow from the client to the server contains almost exclusively ACK packets so that the RTT information is available at the source but not the client.

The objective of the paper is to see whether we can make use of the information about the RTT that is readily is available at the server to design a selection algorithm based on this information. Note that such a server-side algorithm need not replace a client-side control. In fact, we would expect that both ends could work together to improve the streaming control.

### 7.1.1 Adaptive Video Streaming

Video streaming over HTTP previously used the *progressive download method*, in which media players downloaded a simple file from the HTTP web server and they allowed the media to be played back before the entire file has been fully downloaded (Fielding et al., 1999). All clients received the same bitrate despite variations in network condition and client terminals (Fielding et al., 1999). The next wave of HTTP-based video streaming is the *adaptive streaming over HTTP* (26.247, 2015; Stockhammer, 2011) that is now supported by several players such as Adobe Dynamic Streaming, Microsoft IIS Smooth Streaming and Apple HTTP-based Streaming. This technique is known to significantly improve quality by offering a seamless playback experience for users and providing a better reliability and scalability.

Dynamic Adaptive Streaming over HTTP (DASH) was developed by the *3GPP group* (26.247, 2015) and deployed as a solution for adjusting the video bitrate to meet the network condition. It enhances the QoE by dynamically switching quality levels, *i.e., video bitrate*. One recent

approach using this technique is the *Scalable Video Coding* (SVC) where each original video is coded into multiple files with different bitrates. For each quality level, the video content is divided into multiple segments with a playback duration of 2 to 10 seconds (Chen et al., 2016). Following a client request, the HTTP server chooses a segment with the appropriate bit rate to send. At each segment, an adaptive HTTP request-reponse transaction is needed. Recent codecs such as H.264/SVC and MPEG-4/SVC even support bitrate adaptation at the frame level (Chen et al., 2016) in addition to the segment level, which is more flexible for mobile networks. The key feature behind DASH is a congestion control mechanism by letting the user selecting the convenient bitrate to its throughput and stream the appropriate video segment.

### 7.1.2 Related Work

Algorithms for bitrate adaptation for next downloaded video segment can be roughly split into two main groups : Throughput-based and buffer-based.

The throughput-based algorithms rely on network throughput estimation. A linear smoothed throughput prediction is proposed by (Thang et al., 2012; Akhshabi et al., 2012). The decision throughput metric for the $i^{th}$ segment $\hat{A}_i$ is computed using

$$\hat{A}_i = (1 - \delta)\hat{A}_{i-1} + \delta A_i, \tag{7.1}$$

where $A_i$ is the instant throughput for the $i^{th}$ segment and $\delta \in [0, 1]$ is an adaptive parameter depending on the normalized throughput deviation. These algorithms are usually criticized by their late reaction and their lack of sensitivity to other QoE components.

Buffer-based algorithms, on the other hand, which use the occupancy of the jitter buffer at the client side to drive the video level switching, are more aware of network conditions and QoE. These algorithms decide then the bitrate based on the state of the jitter buffer, e.g., through a buffer utilization threshold as proposed in (Zambelli, 2009). The basic idea for this buffer-based algorithm is to determine the buffer state through a jitter buffer controller placed at the client, which periodically sends a feedback to the video server to serve as a decision metric to switch video level. This metric could be the instantaneous jitter buffer length $q(t)$, so that the server reacts by switching down the level when $q(t) > q_o$ and switch up when $q(t) < q_u$. Thresholds $q_o$ and $q_u$ are determined by the media player. This threshold method does not react quickly in case of throughput change, since it must wait for the threshold violation.

A hybrid method was discussed in (Tian and Liu, 2012). It is based on buffer levels and throughput estimation. The rate adaptation controller takes the level switching decision using a *Proportional Integral*, but computing its parameters is difficult in practice.

In fact, recent work (Xu et al., 2014; Le et al., 2013; Chen et al., 2016) has shown how the perceived media quality is basically bound to buffer performance and that that buffer overflow and underflow have the most impact on the viewers' QoE. Overflow, i.e., packet drops, creates video pixelization problem. Underflow, on the other hand, has a significant negative impact on clients' perception since an empty buffer creates video interruptions and a frozen screen. Since video interruptions are the most crucial for users and techniques like error cancellation can cover the packet drop problem in recent codecs technologies, QoE is usually characterized by the buffer underflow probability $P_u$ (Xu et al., 2014). Based on this, the authors of (Chen et al., 2016) propose a new adaptive streaming algorithm based on $P_u$ as an indicator for the jitter buffer state. In this case, the jitter buffer controller is provided with a $P_u$ estimator. An analytic model for estimating $P_u$ is proposed for a Poisson arrival process by modelling the jitter buffer by a M/D/1 queue (Chen et al., 2016). It has been shown that this type of algorithm is stable. However, modeling the buffer arrival traffic by a Poisson process is inaccurate (Paxson and Floyd, 1995; Garcia et al., 2007). In addition, the assumption of an infinite queue is not realistic, especially for mobile terminals.

To summarize, despite the difference between metrics used by these adaptive streaming over HTTP solutions, the common feature is hosting the video level selection algorithm within the client media player. The server always waits for the client request for the next segment. This request contains the selected quality level, which is the main result of these client-based adaptive streaming algorithms. This creates several problems : Firstly, controllers need to make some measurements which are executed by the client terminals. This is energy consuming, especially for mobile clients. Next, these algorithms cause an additional control traffic to communicate client requests to the server. This traffic must be optimally managed. Additionally, the server response may be not prompt enough to cover the instant of the channel change, since it needs to wait for the client-request. Finally, the HTTP protocol returns to the server some information about the round-trip time which could be used to improve control.

### 7.1.3 Contribution

In this paper, we want to investigate whether a server-side control algorithm is feasible for a HTTP video connection and estimate how it can actually react to the network conditions. Our server-side adaptive streaming (SSAS) control belongs to the buffer-based algorithm

group but the decision to change the quality level is moved to the server. This rises two fundamental issues : (1) *How can the server predict the probability of underflow of the client jitter buffer ?* (2) *How far can the SSAS improve QoE ?*

For the first issue, we discuss in section 7.2 how we can estimate the jitter buffer underflow probability $P_u$ from the server. This is based on three elements that we discuss in detail in section 7.2 :

1. There is a strong correlation between the jitter of the traffic entering the jitter buffer and the underflow probability
2. The network can be modelled by an equivalent G/M/1 queue and it is possible to estimate the queue parameters from measurements available at the server and
3. Given the equivalent queue, we can compute quickly the jitter of the traffic.

Item 1 is established by simulation to provide an empirical correlation between the jitter of the stream and the underflow probability. This is different from the work of (Dbira et al., 2016b) in which we established a correlation between the traffic entering the buffer and the underflow probability using a simple arrival distribution. Here, we show similar results but for a distribution of the entering traffic that is actually produced by the network.

Item 2 is based on recent work (Dbira et al., 2017) that explains how the queue parameters can be estimated from measures available to the server, using only TCP acknowledgments (Ack) from the client.

Item 3 uses the results of (Dbira et al., 2016b) to compute the jitter of the traffic being offered to the client buffer. This is different from (Chen et al., 2016) since we use a general distribution for the traffic arrival model and finite buffer model for jitter buffer.

Based on this, we present in section 7.3 the SSAS algorithm, which is based on a server-side network jitter estimation module and on jitter buffer underflow probability. The actual effectiveness is estimated for a small network by simulation in section 7.4 where we see that the SSAS performs well under changing network conditions.

Note again that we do not claim that the server-side algorithm should replace client-based algorithms. In fact, we are convinced that an hybrid model might be developed that could offer the best of both worlds, provide a better control and reduce significantly control traffic and the response time. For now, we concentrate in this paper on showing that a server-side algorithm *can* provide an effective control even when used by itself.

## 7.2 Estimating the Buffer Underflow Probability from the Server

We examine in this section how the server can arrive at a consistent approximation of the jitter buffers' $P_u$. This is based on two main points : First, there is an implicit correlation between network QoS and QoE (Agboma and Liotta, 2008; Alreshoodi and Woods, 2013). Network jitter is an important QoS parameter for video traffic and the same can be said for buffer underflow probability for QoE. Therefore, it is reasonable to think that there is a correlation between network jitter and $P_u$. Second, a server-side estimation of network jitter was proposed by (Dbira et al., 2017). It is based on modeling a video TCP connection between server and client by a G/M/1 queue. The queue parameters are computed using the RTT information available at the server.

### 7.2.1 Relationship between Network Jitter and Buffer Performance

In order to show the benefits of using the network jitter for streaming control, we first study the relationship between network jitter computed analytically for a G/M/1 queue and jitter buffer underflow probability. We recall the network jitter estimation for a TCP connection. Then, we show the relationship between this QoS parameter and $P_u$ using simulation. Finally, we propose a server-side algorithm that can estimate $P_u$ at the client jitter buffer.

#### Server-Side Network Jitter

A widely used definition for network jitter, in particular for video transmission control, is the one given by the Internet Engineering Task Force (IETF) (Cisco, 2007), which is the mean absolute value of the packet delays of two consecutive packets $T_i$ and $T_{i+1}$ from a tagged stream (Demichelis and Chimento, 2002; Angrisani et al., 2013; Poretsky et al., 2006). Jitter $J$ that represents this definition, is then

$$J = E\left[|T_i - T_{i+1}|\right]. \tag{7.2}$$

A computational method to estimate the network jitter for a TCP connection was proposed in (Dbira et al., 2017). After inferring the end-to-end network by a G/M/1 queue for a given TCP stream, it is possible to use an accurate and fast computing model for jitter assessment. The jitter model for this queue equivalent is given by

$$J = \frac{(\eta^2 + \mu^2)}{\eta\mu(\eta + \mu)} + \frac{2}{(\eta + \mu)}\mathcal{L}_{f_R}(\eta + \mu) - \frac{1}{\eta}\mathcal{L}_{f_R}(\eta). \tag{7.3}$$

where

$$\mathcal{L}_{f_R}(s) = \int_0^\infty f_R(y)e^{-sy}dy \tag{7.4}$$

is the Laplace transform of $f_R(y)$ the probability density function of the inter-arrival time variable $R$ for network queue model. The service process $S$ and transit delay $T$ in this G/M/1 queue are exponential by definition and $\mu$ and $\eta$ are respectively the mean service time and mean transit time rates.

The different parameters that intervene in the jitter equation above can very well be estimated at the server side. In fact, the server can select the best $f_R$ distribution that can fit the network arrival. The server can also get the average transit rate $\eta$ from RTT gathered from Acks. Moreover, service rate $\mu$ can be determined by solving a non linear system that bounds transit rate and waiting time probability (Kleinrock, 1975). The algorithm for modeling end-to-end network by a G/M/1 queue and the procedure for estimating jitter numerically are given in (Dbira et al., 2017). The authors in (Dbira et al., 2017) conclude that the network jitter estimation does not depend on the inter-arrival time distribution shape but only on the mean and standard deviation of the $R$ variable. Furthermore, the use of log-normal and gamma distribution for inter-arrival time $R$ can give the best approximation for jitter (more details in the Appendix 7.5).

### Correlation between Analytic Network Jitter and $P_u$

We study in this section the correlation between jitter computed by equation (7.3) that presents network jitter for a video stream transmission and the underflow probability $P_u$. Since the network jitter estimation is computed by the server, our purpose is to show the possibility for the server to get an estimation of $P_u$.

We use simulation to get this relationship. The simulation scenario is sketched in figure 7.1. We model the network as a single First-Come First-Served (FCFS) G/M/1 queue as proposed in (Dbira et al., 2017). Its mean service rate is set at $\mu = 1$. The arrival process to the network is general and it is defined by the mean $m$ and the standard deviation $\sigma$ of the inter-arrival time $R$. A jitter buffer is present at the receiver. Following recommendations from (Dbira et al., 2016b), the jitter buffer is modelled by a G/D/1/K queue operating with a traffic load $\rho_b = 1$. The arrival process to the jitter buffer is defined by the distribution of the packet inter-departure time from the network equivalent queue. Its service rate $\mu_b$ is equal to its arrival rate $\lambda_b$. For these simulations, the jitter buffer size is set to 2 *MBytes* (Li et al., 2013). Thus, if the packet size is 1500 bytes, same as an MTU packet, the buffer capacity $K$ is nearly

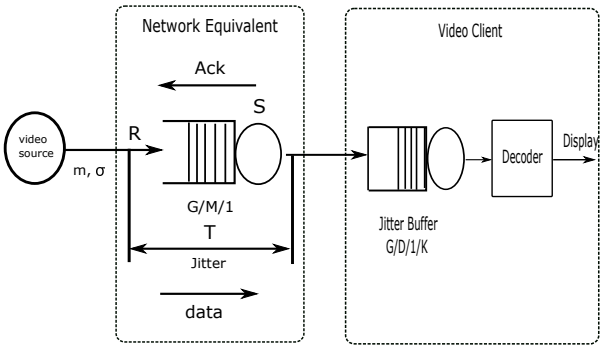



Figure 7.1 Simulation Scenario for Correlation between Network Jitter and Buffer Performance

160 packets. The underflow probability $P_u$ in this buffer is defined as the ratio of buffer server idle time by the total simulation time.

Figures 7.2 to 7.5 show the simulation results expressed in service time unit. We chose two inter-arrival distributions for the network equivalent queue : the gamma distribution and the log-normal distribution. We simulated two cases for network traffic load $\rho \in \{0.4, 0.6\}$. The figures show the jitter at the network equivalent queue, which is numerically computed by equation (7.3), as a function of network arrival standard deviation and the $P_u$ measured in the simulated jitter buffer with respect to the standard deviation of network arrivals.

The reader can note that the correlation is very good for small loads ($\rho = 0.4$) and that is even better when the load increases.

### 7.2.2 Algorithm for a Server-Side $P_u$ Estimation

To better appreciate the correlation between jitter buffer $P_u$ and network jitter, we present in figures 7.6 and 7.7 the same results of figures 7.2 to 7.5 but portrayed with different axis. The

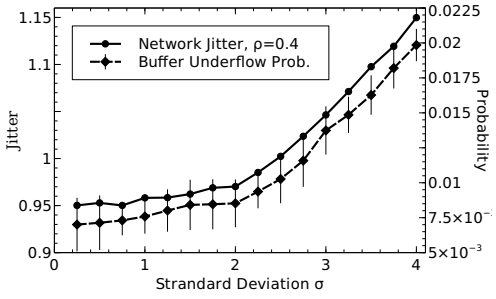



Figure 7.2 Network Jitter and Underflow Probability for Gamma Inter-arrival Time Distribution-$\rho = 0.4$

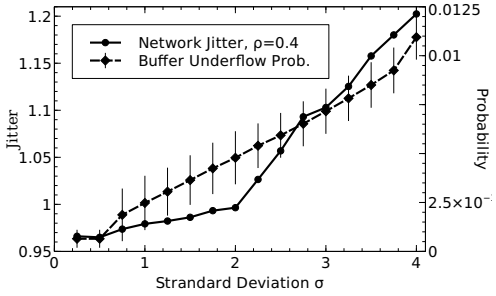



Figure 7.3 Network Jitter and Underflow Probability for Log-normal Inter-arrival Time Distribution-$\rho = 0.4$

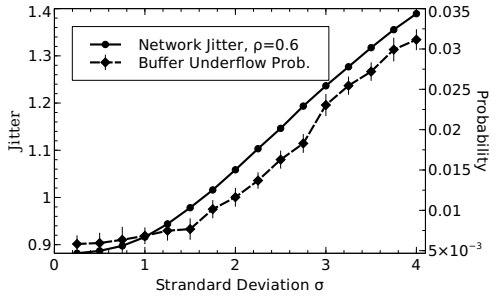



Figure 7.4 Network Jitter and Underflow Probability for Gamma Inter-arrival Time Distribution-$\rho = 0.6$

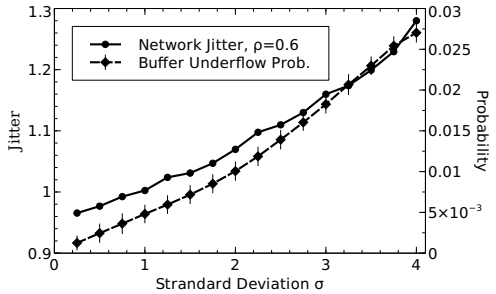



Figure 7.5 Network Jitter and Underflow Probability for Log-normal Inter-arrival Time Distribution-$\rho = 0.6$

relationship between network jitter and $P_u$ at the jitter buffer is slightly linear, in particular for high network traffic load. Using equation (7.3) one can found the jitter knowing the mean and standard deviation of the network inter-arrival time distribution. Then, one can use figures like 7.6 and 7.7 to find the underflow probability.

Based on this, the server-side network jitter estimation could be used as a proxy for buffer performance estimation. This is summarized by the following algorithm :

1. **Initialization** : Curves of the relationship between network jitter $J$ and underflow probability $P_u$ should be pre-stored at the server controller. The algorithm will use only the curve generated for the log-normal distribution. This is because it was found that the log-normal is a good approximation for network arrival process (Dbira et al., 2017). This step is very important to get the mapping between network jitter and $P_u$ values.
2. **Inter-arrival time distribution parameters** : The selected distribution for arrival process is the log-normal distribution (Dbira et al., 2017). Measurement of the mean $m$ and standard deviation $\sigma$ could be identified from the server. These parameters allow to get location $\omega$ and scale $\theta$ related to log-normal distribution using equations (7.8) and (7.9) (see the appendix 7.5).
3. **G/M/1 queue parameters identification** : Mean transit time rate $\eta$ can be estimated from the RTT time computed based on the TCP acknowledgement. As mentioned in (Dbira et al., 2017), service rate $\mu$ could be determined numerically.
4. **Jitter calculation** : Jitter can be evaluated by replacing equation (7.10) (see the appendix 7.5) in equation (7.3).
5. **Calculate underflow probability** : It is possible to approximate $P_u$, given a jitter value, using correlation curves.

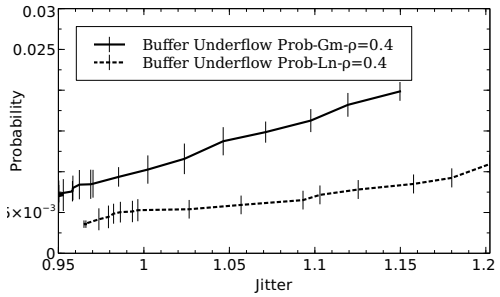



Figure 7.6 Network Jitter Vs Underflow Probability for $\rho = 0.4$

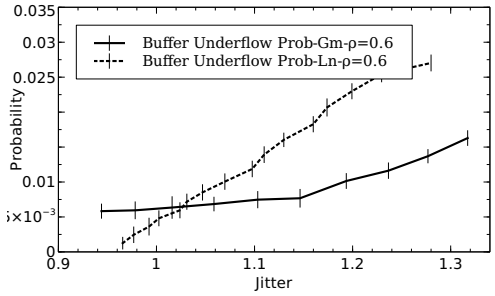



Figure 7.7 Network Jitter Vs Underflow Probability for $\rho = 0.6$

### 7.2.3 Summary

The $P_u$ estimation procedure based on the correlation of this QoE parameter and network jitter shows that the server is becoming able to estimate the buffer state and get an insight on the client QoE. This presents an important result of exploiting the already available information at the server. Thus, the server does not need to wait for feedbacks from the client to get the buffer $P_u$. Furthermore, setting $P_u$ thresholds can be seen as equivalent to setting network jitter thresholds.

## 7.3 Server-Side Adaptive Video Streaming Algorithm

The server-side adaptive video streaming (SSAS) controls the dynamic selection of the video level. Both estimation and decision are made at the video server. We want to check if a server-side algorithm can offer by itself an adaptive streaming control for the client buffer. This is a basic condition to consider in the future a hybrid server-client algorithm for the dynamic adaptive streaming over HTTP.

In this section, we describe the algorithm we propose and we test an implementation by simulation. We show how we use the buffer underflow probability, which is computed based on its correlation with the network jitter, as a decision parameter for the selection of the video level.

### 7.3.1 Overall Architecture

The design of the proposed adaptive streaming includes many components. All units communicate between each other through indicator metrics. Figure 7.8 shows the overall architecture of the SSAS algorithm. The SSAS algorithm is composed by these elements :

1. Network filter,

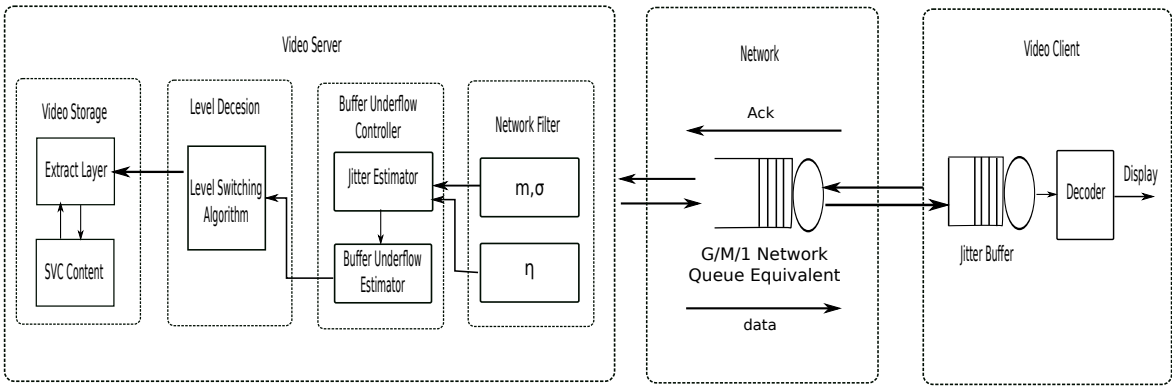



Figure 7.8 Server-Side Adaptive Streaming (SSAS) Algorithm Methodology

2. Buffer underflow controller,
3. Level decision,
4. Video storage.

### 7.3.2 Network Filter

The *Network Filter* collects the parameters available at the server that are used to estimate the network jitter. It measures the mean $m$ and standard deviation $\sigma$ of the inter-arrival time at the network interface for the $N$ packets sent during a given time window $\mathcal{W}$.

$$m = \frac{1}{N}\sum_{i=1}^{N}(t_{i+1}^{s} - t_{i}^{s}) \tag{7.5}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(t_{i+1}^{s} - t_{i}^{s})^2}. \tag{7.6}$$

where $t_i^s$ is the sending instant of the $i^{th}$ packet. The server can also get the RTT from the Ack packets of each TCP connection. From these values, we can approximate the mean transit rate $\eta$ as RTT/2.

### 7.3.3 Buffer Underflow Controller

The main feature of the *Buffer Underflow Controller* unit is to evaluate the network jitter based on the output of the network filter. This is based on the model of section 7.2.1. The module then estimates the jitter buffer underflow probability $P_u$ using the algorithm of section 7.2.2. The calculation is done over a given window size $\mathcal{W}$ , which is taken here as the

length of a video segment. The value computed for $P_u$, called $p$, is sent to the *Level Decision Controller* unit to be compared to some underflow probability thresholds.

### 7.3.4 Level Decision Controller

The *level decision controller* is modelled after the *Google Congestion Control* (GCC) receiver-side controller (De Cicco et al., 2013). This algorithm returns to the server the bitrate for the next packets group through predefined states : increase, decrease or hold. The remote rate controller in the GCC computes the queuing time variation $Q(t_i)$ every time period $t_i$. Based on this information, the algorithm makes a decision about the bitrate state by comparing it to a given threshold $\alpha$. There are three network states possible :

1. *normal* when $|Q(t_i)| < \alpha$.
2. *underused* when $|Q(t_i)| > \alpha$ and $Q(t_i) < 0$.
3. *overused* when $|Q(t_i)| > \alpha$ and $Q(t_i) > 0$.

Here, we adapt the GCC rate controller for use with TCP and in the server. Our controller uses the buffer underflow probability instead of the queuing time variation $Q(t_i)$.

We assume that the quality levels can be classified as a *High Quality* (HQ), *Medium Quality* (MQ) and *Low Quality* (LQ). The algorithm selects the video quality of the next segment $S_{i+1}$ according to the finite state machine given in figure 7.9. We also assume that this unit keeps in memory $l(S_i)$, the level selected for the previous segment $S_i$ and the buffer underflow probability thresholds $\varepsilon$ and $\gamma$.

The model can hold, increase or decrease the video quality level. The controller chooses between those states after receiving the jitter buffer underflow probability $p$ as follows :

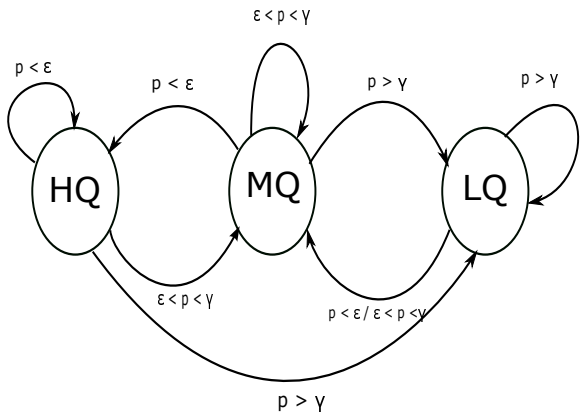



Figure 7.9 Video Quality Level State Machine

1. $p < \varepsilon$ : This means that the buffer occupancy is normal, and the buffer underflow is a rare event. One can improve the QoE by increasing the video quality . If the previous segment $S_i$ was sent with the HQ level then the algorithm holds the same quality for the segment $S_{i+1}$.
2. $\varepsilon < p < \gamma$ : In this scenario, $P_u$ is still reasonably low, but the algorithm tries to protect the buffer from a future underflow. For that reason, if $S_i$ was sent as HQ, the level is decreased by one unit. If it was sent as MQ, the the situation is good enough to keep the same level. However, when the previous segment was sent as LQ, the the level is increased by one unit to keep a good user QoE.
3. $p > \gamma$ : In this situation, $P_u$ exceeds the panic threshold. To keep a good QoE and reduce the video bitrate, the server switches down the level to the LQ if the last segment was sent as either MQ or HQ and holds the same quality if it was sent as LQ.

The decision $\delta \in \{BQ, MQ, LQ\}$ for the level is sent to the video storage unit.

### 7.3.5 Video Storage

The *Video Storage* unit is the video files data base, where each video is segmented and encoded with different quality levels. Depending on $\delta$, the server can select the video file corresponding to the chosen quality and extract the frames corresponding to the next segment.

## 7.4 SSAS System Validation

In this section, we evaluate the feasibility of the SSAS algorithm using simulations. The effectiveness of the algorithm is measured by its ability to minimize the buffer underflow probability $P_u$. We also consider the QoE offered to users through the overall selected levels. We compare the values of $P_u$ produced by the SSAS algorithm with those obtained when the server only sends only high quality video. We also compare the results with those from the MSS algorithm. MSS (Zambelli, 2009) has a receiver-side algorithm that evaluates the buffer occupancy according to a panic $P$ as well as a lower, $L$, and an upper, $U$, threshold. These thresholds are set to 25%, 40% and 80% of the buffer size. When the number of buffered packet is below $P$, the receiver indicates to the server to stream the highest quality. When the buffer occupancy is between $P$ and $L$, the buffer is considered in the steady state and the level is kept unchanged. Otherwise, the bitrate is decreased to the lower quality.

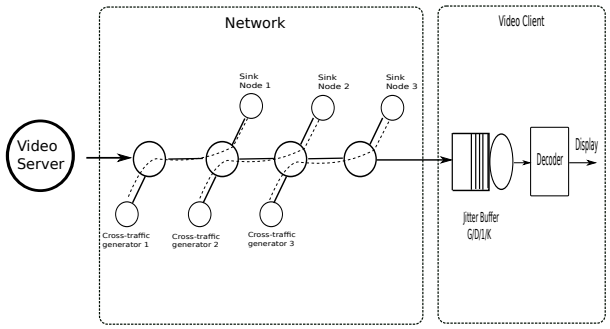



Figure 7.10 Network Simulation for SSAP Test

### 7.4.1 Simulation Overview

We use the simulation scenario in figure 7.10. Three entities are presented in this simulation. First, the video server that hosts the SSAS algorithm is implemented with respect to the entities previously shown in figure 7.8. We use `C++` to implement the algorithm modules. In this simulation, a video sequence encoded in MPEG-4/SVC by the JSVM software package into three quality levels (Seeling et al., 2004) : a base layer and two enhancement layers. These levels are referred to as HQ, MQ and LQ. Additionally, each video sequence is stored in a given file that contains frame sizes. This module returns a frame size file for the ordered segment. Frames pass through the packetizer before being sent to the network. The maximum packet size is set to 1500 bytes.

Next, we find the network which is modelled as a single path of four First-Come First-Served (FCFS) nodes to generate a jittered traffic. All link rates are set to 10 Mbps and we assume that there is a Poisson cross-traffic at each node. The video client entity is made up of the jitter buffer, which is modeled as a G/D/1/K queue. The buffer capacity is set to $K = 160$ packets.

The simulation scenario maintains at first a traffic load $\rho = 0.3$ in all nodes of the network. After sending 30000 packets, we increase abruptly the cross-traffic load in order to congest the network. This is done to observe the impact of the SSAS algorithm on $P_u$.

The simulator calls the SSAS algorithm every $\mathcal{W}$ packets which also determines the video segment length. We assume for this simulation that $\mathcal{W}$ is given in number of packets. We also assume that the server sends the first segment in HQ. Later on, the SSAS algorithm decides the $l(S_{i+1})$ based on its estimate during the current $\mathcal{W}$ period.

### 7.4.2 Simulation Results

Results for the jitter buffer underflow probability and selected levels are given in figures 7.11 to 7.13. We have produced three scenarios with a video segment ($S_i$) size given by $\mathcal{W}$, of 10000, 5000 and 2500 packets. The $\mathcal{W}$ window can also be expressed in seconds : The average frame size can reach 30000 bytes. Thus, we get a segment time length of 4, 8 and 16 seconds, respectively, when the coding rate is 30 frames per second. The figures show the evolution of the buffer underflow under the SSAS and MSS algorithms compared to a HQ streaming without adaptation. We also plot the selected levels for each case where levels 1, 2 an 3 refer to levels HQ, MQ and LQ. We notice the significant decrease of $P_u$ for both the SSAS and MSS algorithms when the network is congested. We find from the curves of figures 7.11b, 7.12b and 7.13b that the SSAS algorithm will use the MQ levels and ramp up and down more gradually than the MSS algorithm, which operates in a somewhat less flexible way, switching between HQ and LQ directly during the congestion period.

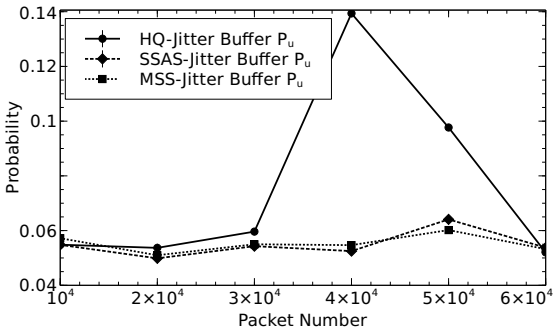
From figure 7.13, one can notice that the SSAS algorithm was faster than the MSS algorithm in detecting congestion and switching down the quality level. This important result is due to the fast jitter increase when the network is congested. This is detected through our algorithm, which anticipates the increase of the jitter buffer underflow.

we notice that the SSAS algorithm is able to stabilize the jitter buffer system during network congestion. The proposed algorithm also shows a good performance in the QoE compared to the client-side algorithm MSS. In most cases and for small video segments, the response of the server was fast compared to the MSS.

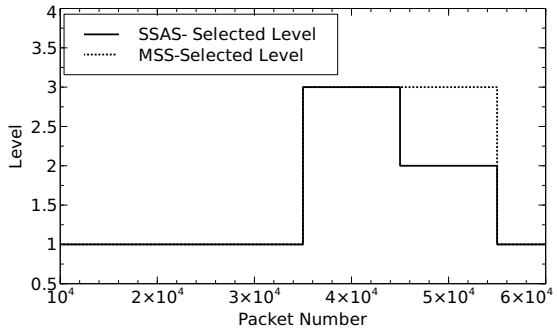
## 7.5 Conclusion

We proposed a new concept based on server side adaptive streaming over HTTP. The algorithm, called SSAS, is based on a computational framework for the jitter buffer underflow probability estimation. It allows the server to dynamically select the video quality level among scalable encoded video segments after estimating the buffer underflow probability through its correlation with network jitter. The server-side jitter model used to estimate the network jitter uses a non-Poisson traffic which is quite accurate for video traffic. A server-side selection algorithms makes use of some readily available information about the network already carried by TCP. It could make decisions faster than from the client side which in turn could reduce the traffic from client requests.

We used simulations to check that the SSAS can reduce the probability of buffer underflow. It can provide a smooth video playback and the offered quality is better than the one for

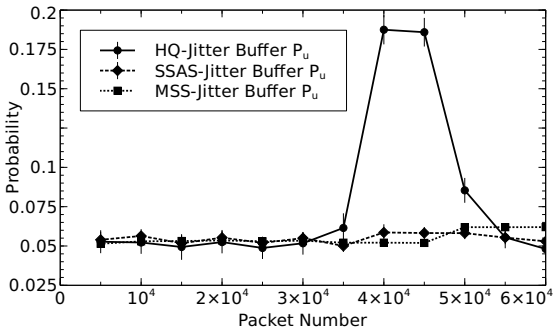



(a) Server-Side Adaptive Streaming Algorithm Effect on Buffer $P_u$

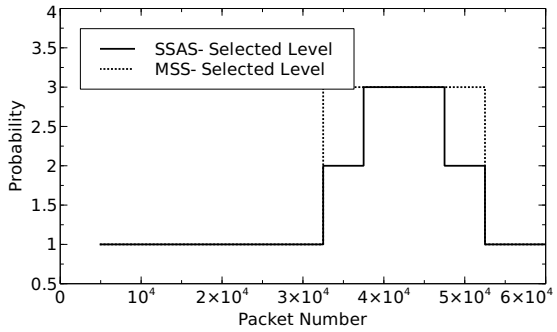



(b) Level Change

Figure 7.11 Results for $\mathcal{W} = 10000$ Packets

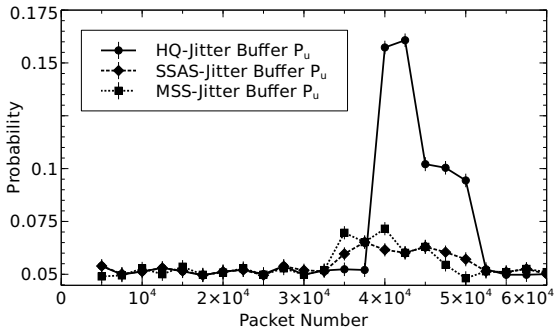



(a) Server-Side Adaptive Streaming Algorithm Effect on Buffer $P_u$

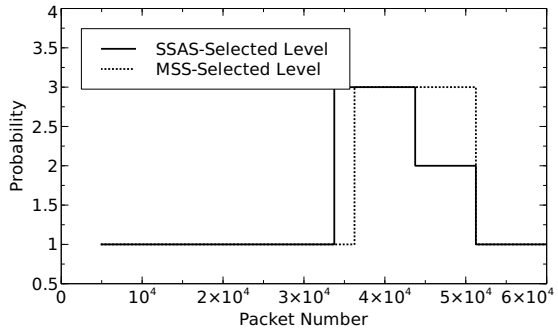



(b) Level Change

Figure 7.12 Results for $\mathcal{W} = 5000$ Packets




(a) Server-Side Adaptive Streaming Algorithm Effect on Buffer $P_u$




(b) Level Change

Figure 7.13 Results for $\mathcal{W} = 2500$ Packets

the client-side algorithm. More importantly, the SSAS algorithm shows a faster reaction to prevent buffer underflow, compared to a client-side algorithm.

And finally, it could be part of an integrated control algorithm operating on both sides of the connection to get a better control. This is because SSAS cannot detect short-term channel changes but it is still a good concept to reduce the load on client terminals and also to reduce control traffic in case of long-term network impairments.

## Appendix A

In case of log-normal distribution, we have

$$f_R(y) = \frac{1}{y\sqrt{2\pi}\theta} \exp\left(-\frac{(\ln y - \omega)^2}{2\theta^2}\right). \tag{7.7}$$

Mean $m$ and standard deviation $\sigma$ of variable $R$ are given in function of location $\omega$ and scale $\theta$ variables.

$$\omega = \ln\left(\frac{m^2}{\sqrt{\sigma^2 + m^2}}\right) \tag{7.8}$$

$$\theta = \sqrt{\ln\left(1 + \frac{\sigma^2}{m^2}\right)} \tag{7.9}$$

So that,

$$\mathcal{L}_{f_R}(s) = \int_0^\infty e^{-sy}\left(\frac{1}{y\sqrt{2\pi}\theta} \exp\left(-\frac{(\ln y - \omega)^2}{2\theta^2}\right)\right). \tag{7.10}$$

Jitter is then computed numerically by using 7.10 in 7.3.

In case of gamma distribution, we have

$$f_R(y) = y^{k-1}\frac{e^{-y/\theta}}{\theta^k \Gamma(k)}. \tag{7.11}$$

Gamma parameters can be written as functions of $m$ and $\sigma$

$$\theta = \frac{\sigma^2}{m}$$
$$k = \left(\frac{m}{\sigma}\right)^2$$

Laplace transform in this case is given by

$$\mathcal{L}_R(s) = \frac{1}{(1+s\theta)^k}. \tag{7.12}$$

From this, we can get then a simplified jitter expression (Dbira et al., 2016a)

$$J = \frac{(\eta^2+\mu^2)}{\eta\mu(\eta+\mu)} + \frac{2}{(\eta+\mu)}\frac{1}{(1+(\eta+\mu)\theta)^k} - \frac{1}{\eta}\frac{1}{(1+\eta\theta)^k}. \tag{7.13}$$

# CHAPITRE 8 DISCUSSION GÉNÉRALE

## 8.1 Synthèse des travaux

Dans cette thèse, nous avons traité différents problèmes :

- Étude analytique de la gigue pour un trafic non-Poisson dans des files d'attente variées.
- Utilisation de la gigue MPPDV pour le contrôle des tampons de gigue pour un flux vidéo.
- Modélisation des connexions vidéo de bout-en-bout avec un modèle G/M/1.
- Application des résultats obtenus pour gérer le trafic de contrôle dans les applications de streaming adaptatives.

Nous avons traité ces sujets avec différentes techniques comme les méthodes analytiques, des approximations, des simulations et des implémentations.

L'objectif du premier axe de recherche était de donner une méthode analytique pour le calcul de la gigue moyenne définie par l'IETF dans dans une seule file d'attente avec un seul flux en utilisant des modèles autres que le modèle du trafic Poisson. Nous avons proposé des modèles analytiques exacts et approximatifs pour estimer ce paramètre important de la QoS. La modélisation repose essentiellement sur une formalisation analytique utilisée auparavant pour le trafic Poisson, ainsi que sur des approximations analytiques pour des cas asymptotiques de la charge de trafic. Dans un premier temps, nous avons exploité un modèle analytique exact pour le calcul de la gigue dans une file d'attente M/M/1 avec un seul flux pour estimer la gigue dans une file d'attente FCFS de type G/M/1. En fait, il était possible de réutiliser ce modèle puisque la distribution du temps de transit dans ce type de file d'attente est exponentielle. Ensuite, une interpolation linéaire a été suggérée pour la gigue en fonction de l'utilisation du lien modélisé par une M/G/1. Cette approximation nécessite seulement l'évaluation analytique ou numérique des points limites qui présentent une charge de trafic nulle ou égale à 1. À la fin, nous avons montré qu'une interpolation en trois segments présente une bonne approximation pour la gigue dans une file d'attente G/D/1. Ces modèles proposés n'exigent pas un calcul complexe et le temps de calcul est pratiquement faible. Ceci nous a mené à notre première contribution : un modèle de calcul pour un trafic généralisé à travers des formules simplifiées et précises. Bien qu'il soit limité au cas d'une seule file d'attente, il peut servir dans divers domaines, comme les tampons de gigue. De plus, il présente un chemin clé vers la formalisation théorique de la gigue sur un réseau de files d'attente.

Le second axe a traité le problème d'application de la gigue pour le contrôle des tampons de gigue. Nous avons montré que la gigue MPPDV est utilisable dans certains cas pour contrôler

les performances des tampons de gigue. Plus précisément, la modélisation du tampon de gigue par une G/D/1/K avec une charge de trafic très élevée, pour avoir un taux d'arrivée égal au taux de service, a montré la corrélation entre la gigue calculée analytiquement ou numériquement et les principales métriques du tampon, la probabilité de débordement et de vidage. La relation entre ces paramètres a conduit à un algorithme d'estimation analytique des probabilités de perte dans ce type de tampon. Ceci pourrait mener à une plateforme analytique pour le contrôle orienté-QoE des tampons de gigue puisque ces probabilités de perte sont prises comme étant les principaux facteurs influençant l'expérience des utilisateurs. Nous pouvons conclure que la mesure de la gigue, qui se base sur la mesure du délai de transit entre l'arrivée et le départ du tampon de gigue, est une bonne application du calcul de la gigue en une seule file d'attente.

En troisième lieu, nous avons étudié la gigue réseau, basé sur l'estimation du délai de transit des paquets du bout-en-bout. La mesure de la gigue dans ce cas est différent de celle de la partie précédente. Les deux points de mesure pour la gigue sont ici entre l'arrivée au réseau du côté du l'émetteur et l'arrivée au client. Nous nous sommes intéressés à une modélisation simplifiés des connexions vidéo de bout-en-bout par une seule file d'attente pour caractériser le trafic vidéo à travers les réseaux Internet. Nous avons supposé une file d'attente G/M/1 comme équivalent du réseau entre l'émetteur et le récepteur de la vidéo. Nous avons proposé un algorithme pour déterminer les caractéristiques de cette file d'attente. Pour ce faire, nous avons suivi une approche expérimentale réalisée avec des connexions TCP et UDP locales ou distantes. L'hypothèse considérée était de négliger les paquets perdus et non ordonnancés. La validation du modèle proposé a impliqué la comparaison de la gigue mesurée pour ces expériences avec la gigue calculée pour une G/M/1. Cette méthodologie est en effet possible puisque la mesure de la gigue n'a pas besoin d'une synchronisation entre les machines. La première étape était de valider l'hypothèse du temps de service exponentiel. Cette étape fondamentale a été démontrée par la distribution des RTTs mesurés pour le trafic TCP. Ensuite, notre procédure de vérification a montré que lorsque le temps d'inter-arrivée à la file G/M/1 suit la log-normale, la gigue expérimentale est la plus proche de celle estimée analytiquement pour la plupart des cas et présente en fait une borne supérieure. Ceci a mené au résultat très important à l'effet que la gigue ne dépend pas très fortement du type de la distribution du temps d'inter-arrivée au réseau et qu'il suffit de déterminer sa moyenne et sa variance. Cette modélisation a introduit une méthode analytique pour estimer la gigue sur le réseau pour une connexion vidéo donnée, en prenant les arrivées comme un processus caractérisé uniquement par ses deux moments.

Enfin, le dernier axe de recherche était consacré à une application de la modélisation analytique de la gigue réseau pour la transmission de la vidéo. L'objectif majeur était d'optimiser

les ressources en adoptant une gestion optimale des informations de contrôle circulant déjà sur le réseau. Cette application est un algorithme de transmission de la vidéo en streaming adaptatif exécuté du coté serveur. Il exploite les données fournies par les acquittements du protocole TCP pour estimer la moyenne du RTT pour ensuite estimer la gigue réseau en se basant sur le modèle de file d'attente G/M/1, puisque la gigue reflète la situation du tampon de gigue à travers les estimés des probabilités de perte. Le déploiement de cet algorithme et son utilisation dans la simulation d'une transmission vidéo adaptative a montré la pertinence de cette approche dans la réduction de la probabilité de vidage du tampon de gigue et donc dans la minimisation des interruptions de la lecture d'une vidéo. La comparaison de ces résultats avec ceux de l'algorithme de *Microsoft Silverlight* MSS, un algorithme du côté client basé sur l'occupation du tampon de gigue, a montré que l'algorithme proposé favorise l'amélioration de la qualité de la vidéo en moyenne par rapport au MSS. Pour résumer, cette dernière partie de notre recherche a démontré une application de l'estimation analytique de la gigue réseau présenté dans la troisième partie du travail.

## 8.2 Limitations des solutions proposées

Au cours de ce projet de recherche, nous avons réalisé de nombreuses limitations des résultats. Nous les présentons brièvement ci-dessous.

Notre modélisation analytique pour la gigue moyenne définie par le standard de l'*IETF* n'est pas applicable pour des transmissions sur IP avec un taux de perte élevé. La définition donnée par MPPDV n'est plus applicable dans ce contexte. En plus, elle ne s'applique pas à des flux multiples multiplexés dans chaque nœud, qui est le cas le plus fréquent dans les réseaux.

Le contrôle dans les tampons de gigue n'est pas instantané. Il n'est réalisable qu'après une période de temps pour que les statistiques de la moyenne et la variance soient significatives. Dans ce cas, notre travail est plus pertinent pour faire les approximations de la probabilité de débordement et de vidage et l'est moins pour faire l'ajustement de sa taille en temps réel. La conception de tout un module de contrôle de l'adaptation des tampons de gigue demeure à faire.

Il n'y a pas encore un moyen d'estimer analytiquement la gigue dans le cas où le tampon de gigue est modélisé par une G/G/1. Ceci présente une modélisation des tampons de gigue lorsqu'il y a ajustement du taux de service adaptatif.

L'estimation proposée pour la gigue dans le réseau est approximative, et repose sur une mesure moyenne de la RTT qui peut être imprécise. De plus, en UDP, elle ne considère que le premier échange de paquets pour estimer le RTT.

La validation expérimentale effectuée pour la capacité du modèle G/M/1 à raccorder avec le réseau de bout-en-bout pour une transmission vidéo peut être insuffisante. Il faut effectuer une étude statistique basée sur un nombre important de mesures faites dans les mêmes conditions du réseau.

Tous les modèles et les applications proposées devraient être vérifiés dans des systèmes non stationnaires.

Nous voulons aussi aborder la modélisation de la gigue en réseau en cas de taux de perte plus élevé.

# CHAPITRE 9 CONCLUSION ET RECOMMENDATIONS

Nous concluons cette thèse consacrée à l'étude analytique de la gigue pour un trafic non-Poisson et ses applications pour le trafic vidéo sur les réseaux Internet, en récapitulant les résultats obtenus et leurs valeurs ajoutées. En outre, nous évoquons d'éventuelles améliorations au travail déjà réalisé.

## 9.1 Synthèse des travaux

Les différents éléments de cette thèse fournissent une base pour l'amélioration de la qualité de service des applications en plein essor sur Internet. par l'étude du paramètre de la gigue, qui présente une mesure qui doit être minimisée pour chaque transmission multimédia, particulièrement pour la vidéo. Il s'agit d'un indicateur de QoS très important, puisqu'elle est directement liée à la qualité perçue par les utilisateurs. Nous avons effectué un travail approfondi pour le calcul de la gigue dans les réseaux IP tout en respectant la simplification et la rapidité des calculs. Nous avons également montré l'utilité de cette mesure pour le contrôle des détériorations dans les tampons de gigue, premiers responsables de la dégradation de la QoE.

Nous avons abordé dans le premier article *Calculation of Packet Jitter for Non-Poisson Traffic* le calcul de la gigue dans le cas d'une seule file d'attente. La littérature fournit des méthodes restreintes pour l'estimation analytique de la gigue dans les réseaux IP. La méthodologie suivie était d'utiliser le processus de Poisson pour la modélisation du trafic IP. Les chercheurs revendiquaient la simplicité et la flexibilité du modèle malgré son échec à modéliser le trafic IP réel. En pratique, la gigue est calculée sur un flux de contrôle qui n'est pas toujours précis à cause de la synchronisation. La contribution de cette première partie du travail réside dans l'étude de la gigue en cas d'un trafic non-Poisson, ce qui généralise les travaux antérieurs. Nous avons en plus respecté l'aspect de la simplicité des modèles de calcul proposés. Il s'agit d'une modélisation qui permet de réduire le temps de calcul.

Les résultats du premier article ont servi au deuxième article *On the Relationship between Packet Jitter and Buffer Loss Probabilities*, où le tampon de gigue est modélisé comme une seule file d'attente. En effet, le calcul de la gigue est basé sur la définition donnée par le MPPDV proposée par l'IETF. Cette définition était souvent considérée comme inadéquate pour le dimensionnement des tampons de gigue. En outre, les modèles proposés sont construits pour des files d'attente infinies, ce qui n'est pas le cas pour les tampons de gigue. Notre contribution

a été de montrer la possibilité de contrôler le tampon de gigue avec l'estimation analytique de la gigue MPPDV. Tout d'abord, nous avons montré que dans un pareil cas, il est possible d'utiliser le modèle de l'estimation de gigue en cas d'une G/D/1/$\infty$ pour les tampons de gigue. Ensuite, nous avons proposé un algorithme qui estime les performances du tampon de gigue analytiquement. Une façon de prolonger ce travail est de concevoir un algorithme qui fait le contrôle du tampon de gigue en se basant sur des paramètres de la QoE estimés analytiquement à partir de la gigue.

L'objectif du troisième article *End-to-End Network Queuing Model Equivalent for Video Applications* est la modélisation du réseau pour une transmission vidéo. Nous avons montré qu'une modélisation de réseau en G/M/1 est adéquate pour le flux vidéo. Ceci fournit un résultat très important : une méthode de calcul de la gigue en réseau du coté serveur. Encore une fois nous faisons appel aux résultats du premier article. Ce travail est applicable pour les transmissions vidéo, surtout transportés par TCP. Dans le cas de l'UDP, le calcul est approximatif, puisqu'il y a absence de trafic de contrôle pour déterminer le RTT exact.

Le résultat de la troisième partie de la thèse est utilisé dans le quatrième article *A Server-Side Adaptive Control for Video Streaming*. Nous y exploitons la nouvelle méthode d'estimation de gigue réseau du coté serveur pour une transmission vidéo afin de déterminer la performance du tampon de gigue du coté client. L'idée était principalement d'utiliser la probabilité de vidage (*Underflow*) comme le critère d'adaptation de la qualité de la vidéo envoyée par le serveur. Actuellement, la sélection du débit de l'envoi se fait par le client et l'information est transféré par la suite au serveur. Notre contribution réside dans la discussion de la possibilité d'exploiter les informations disponibles pour le serveur pour faire ce contrôle. Cela va diminuer significativement les requêtes de contrôle qui circulent sur le réseau pour assurer la bonne transmission de la vidéo.

## 9.2 Améliorations futures

Les possibilités de développement de nos travaux sont nombreuses, principalement dans l'étude analytique de la gigue dans un réseau de files d'attente, les applications et les implémentations.

Il serait utile de trouver un moyen pour estimer la gigue de bout-en-bout d'un trafic marqué qui passe à travers $N$ nœuds en série, comme dans l'exemple de la fig. 9.1. Il faut noter que pour deux nœuds en tandem, le délai de séjour du paquet $i$ dans le second nœud dépend de son temps de service dans le premier nœud. Cette corrélation a aussi un effet sur la gigue dans le second nœud et donc sur la gigue de bout-en-bout.

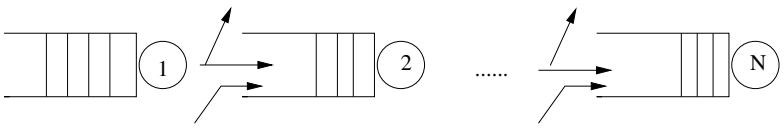



Figure 9.1 Exemple de $N$ nœuds en tandem

Par ailleurs, la gigue algébrique sur un intervalle de temps $\Delta t$ est un estimé de l'évolution du délai de transmission dans le temps $t$. Dans un modèle fluide en régime non-stationnaire, nous avons alors

$$J = \frac{dT}{dt}dt. \tag{9.1}$$

Puisque le délai moyen $T(t)$ est peut-être relié aux taux d'arrivée, il serait possible de concevoir une module de contrôle des tampons de gigue qui prévoit analytiquement la direction de l'occupation du tampon vers un débordement ou vers un vidage.

Un travail futur très pertinent serait l'implémentation d'un protocole pour le contrôle des performances du tampon de gigue et l'évaluation de son fonctionnement et, surtout, de son temps de réponse. On peut l'intégrer dans un algorithme de contrôle basé sur l'estimation de la QoE qui doit être comparé à d'autres algorithmes existants.

Une tendance chez les grands fournisseurs d'équipements de télécommunication, comme Cisco et Juniper, est de fournir des routeurs qui font les mesures des paramètres de la QoS par eux-même, afin de les utiliser dans un routage orienté qualité de service. Dans cette perspective, une estimation analytique peut être intéressante en réduisant le trafic de mesure qui circule sur le réseau. Par ailleurs, la formalisation analytique de la gigue ouvre des perspectives sur l'optimisation du routage pour le trafic vidéo basé sur la minimisation de la gigue.

Avec des méthodes de calcul simples et précises, on peut utiliser la gigue dans plusieurs autres domaines. Comme une intégration de la gigue comme un indicateur dans les scheduleurs des ressources radio en LTE ou dans les ressources virtuelles des centres de données, puisque la gigue est très nuisible pour certaines applications comme les jeux en ligne et dans le e-commerce boursier.

# LISTE DES RÉFÉRENCES

"Recommendation Y.1541 : Network performance objectives for IP-based services", International Telecommunications Union, Fév. 2006.

"Internet protocol data communication service—IP packet transfer and availability performance parameters", ITU-T Recommendation Y.1540, Mars 2011. En ligne : `https://www.itu.int/rec/T-REC-Y.1540-201103-I/en`

G. T. 26.247, "Transparent end-to-end packet-switched streaming service (pss) ; progressive download and dynamic adaptive streaming over http (3gp-dash)", Jan. 2015.

F. Agboma et A. Liotta, "QoE-aware QoS management", dans *Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia*. ACM, 2008, pp. 111–116.

S. Akhshabi, S. Narayanaswamy, A. Begen, et C. Dovrolis, "An experimental evaluation of rate-adaptive video players over HTTP", *Signal Processing : Image Communication*, vol. 27, no. 4, pp. 271–287, 2012.

S. Alouf, P. Nain, et D. Towsley, "Inferring network characteristics via moment-based estimators", dans *Proc. INFOCOM*, vol. 2, Avr. 2001, pp. 1045–1054. DOI : `10.1109/INFCOM.2001.916298`

M. Alreshoodi et J. Woods, "Survey on QoE\ QoS correlation models for multimedia services", *International Journal of Distributed and Parallel Systems (IJDPS)*, vol. 4, Mai 2013.

H. Alshaer et J. Elmirghani, "Expedited forwarding end-to-end delay and jitter in DiffServ", *International Journal of Communication Systems*, vol. 21, pp. 815–841, 2008.

L. Angrisani, D. Capriglione, L. Ferrigno, et G. Miele, "An Internet Protocol packet delay variation estimator for reliable quality assessment of video-streaming services", *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 5, pp. 914–923, 2013.

G. Apostolopoulos, R. Guérin, S. Kamat, et S. Tripathi, "Quality of service based routing : a performance perspective", dans *Proceedings of the ACM SIGCOMM'98*, Vancouver, Canada, Sep. 1998.

D. Awduche, “MPLS and traffic engineering in IP networks”, *IEEE Communications Magazine*, vol. 37, no. 12, pp. 42–47, Déc. 1999.

G. Barberis et D. Pazzaglia, “Analysis, design and buffer sizing of a packet voice receiver”, *IEEE Transactions on Communincations*, vol. 28, pp. 217–27, 1980.

P. Barreto et P. Carvalho, “Network planning optimization for multimedia networks”, dans *Proc. 8th IEEE International Symposium on Network Computing and Applications*, Juil. 2008, pp. 60–67.

T. Benson, A. Akella, et D. Maltz, “Network traffic characteristics of data centers in the wild”, dans *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 267–280.

R. Braden, “Transmission control protocol”, IETF RFC 793, Sep. 1981.

O. Brun, C. Bockstal, et J. Garcia, “Analytical approximation of the jitter incurred by CBR traffics in IP networks”, *Telecommunication Systems*, vol. 33, pp. 23–45, Déc. 2006.

S. Chen et K. Nahrstedt, “An overview of quality of service routing for next generation high speed networks”, *IEEE Network*, vol. 12, pp. 64–79, 1998.

S. Chen, J. Yang, Y. Ran, et E. Yang, “Adaptive layer switching algorithm based on buffer underflow probability for scalable video streaming over wireless networks”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 6, pp. 1146–1160, 2016.

J.-M. Chung et H.-M. Soo, “Jitter analysis of homogeneous traffic in differentiated services networks”, *IEEE Communications Letters*, vol. 7, no. 3, pp. 130–132, 2003. DOI : 10.1109/LCOMM.2003.809987

C. Cifliklia, A. Gezera, A. Özşahina, et O. Özkasapb, “Bittorrent packet traffic features over IPv6 and IPv4”, *Simulation Modelling Practice and Theory*, vol. 18, pp. 1214–1224, Oct. 2010.

Cisco, “IP service level agreement”, 2007.

“Cisco visual networking index : Forecast and methodology, 2014-2019 white paper”, Mai 2015. En ligne : http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html

A. Clark, “Analysis, measurement and modeling of jitter”, Telchemy Incorporated, USA, Rapp. tech., Jan. 2003. En ligne : https://www.telchemy.com/reference/ITUSG12_JitterAnalysis.pdf

A. Clark et G. Claise, “RFC 6390 : Guidelines for considering new performance metric development”, IETF, Oct. 2011.

A. Clark et Q. Wu, “RFC 6798 : RTP control protocol (RTCP) extended report (XR) block for packet delay variation metric reporting”, IETF, Nov. 2012.

M. Coates et R. Nowak, “Network loss inference using unicast end-to-end measurement”, dans *Proc. ITC Conference on IP Traffic, Modeling and Management*, 2000, pp. 28–1.

M. E. Crovella et A. Bestavros, “Self-similarity in World Wide Web traffic : Evidence and possible causes”, *IEEE/ACM transaction on Networking*, vol. 5, no. 6, pp. 835–846, Déc. 1997.

H. Dahmouni, D. Rosse, B. Morin, et S. Vaton, “Impact of data traffic composition on GPRS performance”, dans *Proc. 19th International Teletraffic Congress*, Beijing, Août 2005.

H. Dahmouni, A. Girard, M. Ouzineb, et B. Sansò, “The impact of jitter on traffic flow optimization in communication networks”, *IEEE Transactions on Network and Service Management*, vol. 9, no. 3, pp. 279–292, Sep. 2012.

H. Dahmouni, A. Girard, et B. Sansò, “An analytical model for jitter in IP networks”, *Annals of Telecommunications*, vol. 67, pp. 81–90, Jan. 2012. En ligne : http://www.springerlink.com/content/5046104335872460/

H. Dbira, A. Girard, et B. Sansò, “Calculation of packet jitter for non-Poisson traffic”, Gerad, Rapp. tech. G-2014-92, Mars 2014. En ligne : https://www.gerad.ca/fr/papers/G-2014-92/view

——, “Calculation of packet jitter for non-Poisson traffic”, *Annals of Telecommunication*, vol. 71, no. 2, pp. 223–237, Mai 2016. DOI : 10.1007/s12243-016-0492-0

H. Dbira, A. Girard, et B. Sansò, “On the relationship between packet jitter and buffer loss probabilities”, dans *Proc. International Network Strategy and Planning Symposium*, Sep. 2016.

——, “End-to-end network queuing model equivalent for video applications”, Fév. 2017, submitted to Computer Networks. En ligne : https://www.gerad.ca/fr/papers/

G-2016-85/view

L. De Cicco, G. Carlucci, et S. Mascolo, “Experimental investigation of the Google congestion control for real-time flows”, dans *Proc. of the ACM SIGCOMM Workshop on Future Human-Centric Multimedia Networking*, Août 2013, pp. 21–26. DOI : 10.1145/2491172.2491182. En ligne : http://doi.acm.org/10.1145/2491172.2491182

C. Demichelis et P. Chimento, “RFC 3393 : IP packet delay variation metric for IP performance metrics (IPPM)”, IETF, Nov. 2002.

R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, et T. Berners-Lee, “Hypertext transfer protocol – HTTP/1.1”, IETF RFC 2616, Juin 1999.

M. Fischer et D. Masi, “Analyzing Internet packet traces using Lindley’s recursion”, dans *Proc. Winter Simulation Conference*, 2006, pp. 2195–2201.

C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, et S. Diot, “Packet-level traffic measurements from the Sprint IP backbone”, *Network*, vol. 17, no. 6, pp. 6–16, 2003.

R. Garcia, X. Paneda, V. Garcia, D. Melendi, et M. Vilas, “Statiscal characterization of a real video on demand service : User behaviour and streaming-media workload analysis”, *Simulation Modelling Practice and Theory*, vol. 15, no. 6, pp. 672–689, 2007.

I. Gradshteyn et I. Ryzhik, *Table of Integrals, Series, and Products*, 7e éd. Elsevier Inc., 2007.

L. Hanzo, P. Cherriman, et J. Streit, *Video compression and communications*, 2e éd. WILEY, 2007.

D. Hayman, “Sizing backbone internet links”, *Operations Research*, vol. 53, pp. 575–585, Août 2005.

T. Hoang, “Planning and optimization of multi-service computer networks”, dans *Proc. 10th Communications and Networking Simulation Symposium*, Mars 2007.

N. Hu et P. Steenkiste, “Evaluation and characterization of available bandwidth probing techniques”, *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 6, pp. 879–894, Août 2003. DOI : 10.1109/JSAC.2003.814505

A. Huremovic et M. Hadzialic, “Novel approach to analytical jitter modeling”, *Journal of Communications and Networks*, vol. 17, no. 5, pp. 534–540, 2015.

C. V. N. Index, “Total ip traffic to grow 3x from 2012 to 2017”, Cisco, White Paper, Mai 2012.

——, “The zettabyte era—trends and analysis”, Cisco, White Paper, Juin 2016.

V. Jacobson, R. Braden, et D. Borman, “RCP extensions for high performance”, IETF RFC 1323, Mai 1992.

A. Kansar et A. Karandikar, “Jitter-free audio playout over best effort packet networks”, dans *ATM Forum International Symposium, New Delhi, India*. Citeseer, 2001, pp. 224–231.

P. Kelly et P. Key, “Dimensioning playout buffers from an ATM network”, dans *Proc 11th Teletraffic Symposium*. IET, 1994, pp. 16A–1.

L. Kleinrock, *Queuing Systems*. Wiley, 1975.

H. Kobayashi et A. Konheim, “Queueing models for computer communications system analysis”, *IEEE Transactions on Communications*, vol. 25, no. 1, pp. 2–29, 1977.

S. Krishnan et R. Sitaraman, “Video stream quality impacts viewer behavior : inferring causality using quasi-experimental designs”, *IEEE/ACM Transactions on Networking*, vol. 21, no. 6, pp. 2001–2014, 2013.

H. Le, D.V., N. N. Ngoc, A. Pham, et T. Thang, “Buffer-based bitrate adaptation for adaptive HTTP streaming”, dans *2013 International Conference on Advanced Technologies for Communications (ATC 2013)*. IEEE, 2013, pp. 33–38.

W. Leland, M. Taqqu, W. Willinger, et D. Wilson, “On the self-similar nature of Ethernet traffic”, *IEEE/ACM Transactions on Networking*, pp. 1–15, Fév. 1994.

M. Li, “QoE-Based performance evaluation for adaptive media playout systems”, *Advances in Multimedia*, vol. 2013, Jan. 2013. En ligne : `http://dx.doi.org/10.1155/2013/152359`

M. Li, T.-W. Lin, et S.-H. Cheng, “Arrival process-controlled adaptive media playout with multiple thresholds for video streaming”, *Multimedia Systems*, vol. 18, pp. 391–407, 2012. DOI : `10.1007/s00530-012-0260-6`

X. Li, Z. Qian, S. Lu, et J. Wu, “Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center”, *Mathematical and Computer Modelling*, vol. 58, no. 5-6, pp. 1222–1235, sep 2013. DOI : 10.1016/j.mcm.2013.02.003

L. Liang, Y. Chen, et Z. Sun, “Characterisation of internet traffic in wireless networks”, *Network Performance Engineering*, vol. 5233, pp. 191–202, 2011. En ligne : http://link.springer.com/chapter/10.1007/978-3-642-02742-0_9#

H. Magri, N. Abghour, et M. Ouzzif, “Analytical models for jitter and QoS requirements with IPP and MMPP-2 traffics”, dans *International Conference on Wireless Networks and Mobile Communications*. IEEE, 2015, pp. 1–6.

M. Mandjes, K. van der Waland, K. Rob, et H. Bastiaansen, “End-to-end delay models for interactive services on a large-scale IP network”, dans *Proceedings of the 7th workshop on performance modelling and evaluation of ATM & IP networks*, 1999, pp. 28–30.

W. Matragi, C. Bisdikian, et K. Sohraby, “Jitter calculus in ATM networks : single node case”, dans *Proc. IEEE INFOCOM*, vol. 1, Toronto, 1994, pp. 232–241.

W. Matragi, K. Sohraby, et C. Bisdikian, “Jitter calculus in ATM networks : Multiple node case”, *IEEE/ACM Transactions on Networking*, vol. 5, pp. 122–133, 1997.

C. L. Ming et J. Schormans, “Measurement-based end to end latency performance prediction for SLA verification”, *Journal of Systems and Software*, vol. 74, no. 3, pp. 243–254, 2005.

A. Morton et G. Claise, “RFC 5481 : Packet delay variation applicability statement”, IETF, Mars 2009.

L. Muscariello, M. Mellia, M. Meo, M. Ajmone Marsan, et R. Lo Cigno, “An MMPP-based hierarchical model of internet traffic”, dans *IEEE International Conference on Communications*, 2004, pp. 2143–2147.

S. Nadarajah et S. Kotz, “On the laplace transform of the pareto distribution”, *IEEE Communications Letters*, vol. 10, no. 9, p. 682, Sep. 2006.

NagyM., V. Singh, J. Ott, et L. Eggert, “Congestion control using fec for conversational multimedia communication”, dans *Proceedings of the 5th ACM Multimedia Systems Conference*. ACM, 2014, pp. 191–202.

M. Narbutt et L. Murphy, “Improving voice over IP subjective call quality”, *IEEE Communications Letters*, vol. 8, no. 5, pp. 308–310, 2004.

N. Omnes, A. Gravey, R. Marie, et B. Sericola, “Mathematical modeling of QoS for ATM CBR services”, dans *Proceedings of the 13th International Symposium on Computer and Information Sciences*, vol. 53. IOS Press, Oct. 1998, p. 57.

A. ParandehGheibi, M. Médard, A. Ozdaglar, et S. Shakkotai, “Avoiding interruptions—a QoE reliability function for streaming media applications”, *IEEE Journal on Selected Areas in communications*, vol. 29, no. 5, pp. 1064–1074, Mai 2011.

V. Paxson et S. Floyd, “Wide area traffic : The failure of Poisson modeling”, *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, Juin 1995.

S. Poretsky, J. Perser, S. Erramilli, et S. Khurana, “Terminology for benchmarking network-layer traffic control mechanisms”, IETF RFC 4689, Oct. 2006.

J. Postel, “User datagram protocol”, IETF RFC 7768, Août 1980.

A. Privalov et K. Sohraby, “Per-stream jitter analysis in CBR ATM multiplexors”, *IEEE/ACM Transactions on Networking*, vol. 6, pp. 141–149, 1998.

R. Ramjee, J. Kurose, D. Towsley, et H. Schulzrinne, “Adaptative playout mechanisms for packetized audio applications in wide-area networks”, dans *Proc. Infocom*, vol. 2, 1994, pp. 680–688.

J. Roberts et F. Guillemin, “Jitter in ATM networks and its impact on peak rate enforcement”, *IEEE/ACM Transactions on Networking*, vol. 16, pp. 35–48, 1992.

E. Rolland, A. Amiri, et R. Barkhi, “Queueing delay guarantees in bandwidth packing”, *Computers and operations research*, vol. 26, no. 9, pp. 921–935, 1999.

D. Rossi, M. Mellia, et M. Meo, “Understanding skype signaling”, *Computer Networks*, vol. 53, no. 2, pp. 130–140, 2009.

B. Sansò et P. Soriano, éds., *Telecommunications Network Planning*, série Center for Transportation Research 25þth Anniversary Series. Norwell, Massachusetts : Kluwer Academic Publishers, 1998.

H. Schulzrinne, S. Casner, R. Frederick, et V. Jacobson, “RFC 3550 : RTP : A transport protocol for real-time applications”, IETF, Juil. 2003.

P. Seeling, M. Reisslein, et B. Kulapala, “Network performance evaluation using frame size and quality traces of single-layer and two-layer video : A tutorial”, *IEEE Communications*

*Surveys & Tutorials*, vol. 6, no. 3, pp. 58–78, 2004.

J. Seo, V. Leung, et H. Lee, "Optimizing a playout buffer with queueing performance metrics for one-way streaming video", dans *Proc. Globecom*. IEEE, Nov. 2008, pp. 1–6.

P. Skurowski, R. Wójcicki, et Z. Jerzak, "Evaluation of IP transmission jitter estimators using One-Way Active Measurement Protocol (OWAMP)", *Communications in Computer and Information Science*, vol. 79, pp. 153–162, 2010.

C. Sreenan, J. Chen, P. Agrawal, et B. Narendran, "Delay reduction techniques for playout buffering", *IEEE Transactions on Multimedia*, vol. 2, no. 2, pp. 88–100, 2000.

S. Srivastava, A. V. de Liefvoort, et D. Medhi, "Traffic engineering of MPLS backbone networks in the presence of heterogeneous streams", *Computer Networks*, vol. 53, pp. 2688–2702, 2009.

T. Stockhammer, "Dynamic adaptive streaming over HTTP : standards and design principles", dans *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 133–144.

C. Tellambura et D. Senaratne, "Accurate computation of the MGF of the lognormal distribution and its application to sum of lognormals", *IEEE Transactions on Communications*, vol. 58, pp. 1568—-1577, 2010.

T. Thang, Q. Ho, J. Kang, et A. Pham, "Adaptive streaming of audiovisual content using mpeg dash", *IEEE Transactions on Consumer Electronics*, vol. 58, no. 1, pp. 78–85, 2012.

G. Tian et Y. Liu, "Towards agile and smooth video adaptation in dynamic HTTP streaming", dans *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. ACM, 2012, pp. 109–120.

W. Verhelst et M. Roelands, "An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech", dans *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, vol. 2. IEEE, 1993, pp. 554–557.

Y. Xu, E. Altman, R. El-Azouzi, M. Haddad, S. Elayoubi, et T. Jimenez, "Analysis of buffer stravation with application to objective QoE optimization of streaming services", *IEEE Transactions on Multimedia*, vol. 16, pp. 813–827, Avr. 2014.

A. Zambelli, “IIS smooth streaming technical overview”, *Microsoft Corporation*, vol. 3, p. 40, 2009.